VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Měření teplot pomocí programovatelného automatu B&R a teplotních čidel připojených na sběrnici 1-Wire

Temperature Measurement by Programmable Controller B&R and Sensors Connected Using 1-Wire bus

2010 **Bc. Tomáš Vavrla**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Zadání diplomové práce

Student: **Bc. Tomáš Vavrla**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Měření teplot pomocí programovatelného automatu BaR a teplotních čidel připojených na sběrnici 1-wire
Temperature Measurement by Programmable Controller B&R and Sensors Connected Using 1-wire bus

Zásady pro vypracování:

1. Rozbor současného stavu problematiky měření teplot ve vrtech výzkumného polygonu na VŠB-TU Ostrava.
2. Rozbor problematiky měření fyzikálních veličin snímači připojenými na sběrnici 1-Wire.
4. Návrh a realizace knihovny pro měření teplot pomocí snímačů s rozhraním 1-Wire pro programovatelné automaty B&R.
5. Testování knihovny v laboratorním a reálném provozu.
6. Návrh a realizace vizualizační aplikace pro nastavení parametrů měření.
7. Zhodnocení výsledků.

Seznam doporučené odborné literatury:

1. KOZIOREK, Jiří. Programovatelné automaty a vizualizace řídicích systémů. Výukový text pro studenty oboru Měřicí a řídicí technika, VŠB – TU Ostrava, 2009.
2. Firemní technická dokumentace pro použité komponenty řídicího a vizualzačního systému.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 6.5.2011 Podpis studenta

Bc. Tomáš Vavrla

Poděkování

Za účinnou podporu, cenné připomínky a rady při zpracovávání diplomové práce tímto děkuji panu Ing. Petru Vojčinákovi a vedoucímu diplomové práce panu Doc. Ing. Jiřímu Koziorkovi, Ph.D. Dále chci poděkovat svým rodičům za podporu při studiu na vysoké škole.

Abstrakt

Tato diplomová práce se zabývá vývojem knihovny a vizualizace pro měření teploty pomocí sběrnice 1-Wire a programovatelného automatu B&R Systém X20.

Práce popisuje vývoj knihovny a její implementaci do programovatelného automatu v reálném a testovacím provozu.

Pro testování knihovny v laboratorních podmínkách je zvlášť vytvořen program, který běží na klasickém stolním počítači. Program vypisuje veškeré informace přijaté z rozhraní 1-Wire do okna programu.

Abstract

This thesis deals with the development of libraries and visualization for temperature measurement using 1-Wire bus and a programmable controller B & R X20 System.

The work describes the development of the library and its implementation into PLC.

The work describes the development of the library and its implementation into a programmable controller and test in real traffic.

To test the library in the laboratory is specifically created a program that runs on a classical computer. The program lists all of the information received from the 1-Wire in the window.

Klíčová slova

Automation Studio, Promotic, teplotní čidlo DS18B20, převodník ADA-101W, PLC B&R Systém X20

Key Words

Automation Studio, Promotic, temperature sensor DS18B20, ADA-101W Converter, PLC B&R System X20

Seznam použitých symbolů a zkratek

ADC – Analog Digital Converter – Analogově číslicový převodník
B&R – Berneker & Rainer
Buffer – Vyrovnávací pole nebo paměť
CAN – Controller Area Network – Sériová sběrnice komunikující na bázi protokolu CAN
CMOS – Complementary Metal Oxide Semiconductor – Komplementární technologie kov nekov polokov
CPU – Central Processor Unit – Centrální procesorová jednotka
CRC – Cyclic Redundant Control – Cyklická redundantní kontrola
DQ – Datový pin převodníku ADA-101W
EEPROM – Electronically Erasable Programmable Read-Only Memory – Elektricky mazatelná a paměť typu ROM-RAM
GND – Ground – Nulový potenciál pro uzemnění
HMI/SCADA – Human Machine Interface/Supervisory Control And Data Acquisition – Rozhraní člověk stroj/Dispečerské řízení a získávání dat – Vizualizace systému
ID – Identifikační číslo
LED – Light Emitted Diode – Dioda vyzařující světlo
Logická 0 – Napětí o velikosti 0 až 0,2 Ucc [V]
Logická 1 – Napětí vyšší než 0,5 Ucc [V]
OPC – Object Linking and Embedding for Process Control – Objektové řízení procesů v reálném čase
PLC – Programmable Logic Controller – Programovatelný automat
RAM – Random Access Memory – Paměť s libovolným přístupem
ROM kód – 64 bitové identifikační číslo 1-Wire zařízení
RS232 – Počítačové sériové komunikační rozhraní
SRAM – Static Random Access Memory – Statická paměť s libovolným přístupem
TTL – Transistor Transistor Logic – Tranzistor Tranzistorová logika
USB – Universal Serial Bus – Univerzální sériová sběrnice
Utilita – Pomocné programy nebo funkce
VDD – Pin kladného napětí pro sběrnici 1-Wire

Obsah

1 Úvod

V dnešní době si lidstvo začíná uvědomovat otázku hledání alternativních zdrojů energie, neboť éra fosilních paliv se blíží ke konci. Geotermální energie je jednou z mnoha možných typů energií určených k získání čisté energie. Pojmem "geotermální energie" je možné v širším pojetí definovat jako teplo obsažené v zemském jádru a plášti. Využívá se ve formě tepelné energie pro vytápění bytů, či pro výrobu elektrické energie v geotermálních elektrárnách. Využití této energie je sice na první pohled jednoduché, ale technické řešení je náročnější, protože horká voda získána ze samotného vrtu obsahuje mnoho minerálů a tím zanáší technologická zařízení. Z toho vyplívá častá údržba a čištění celého systému.

Tato diplomová práce je zaměřena na rozvoj měření teploty měřícího systému tepelných vrtů. Tímto rozvojem je myšleno zdokonalení stávajícího systému na nový. Nový systém bude obsahovat PLC automat firmy Bernecker & Rainer, který bude obsahovat vytvářenou knihovnu pro získávání informací o teplotě v jednotlivých částech vrtu. Co se týče vizualizace v Promoticu bude vytvořena aplikace pro sledování měření a nastavování parametrů měření.

Základem práce je znalost stávajícího systému pro měření teploty tepelných vrtů. Tímto rozborem se zabývá druhá kapitola. Součástí druhé kapitoly je i rozbor parametrů použitých zařízení. Mezi tyto zařízení patří PLC automat a 1-Wire senzor teploty. Důležitá je i znalost mikroLAN sítě a 1-Wire protokolu. Tímto rozborem se zabývá třetí kapitola. Čtvrtá kapitola se zaměřuje na realizaci měřícího systému ze stávajícího systému. Součástí této kapitoly je i popis topologie sítí a popis převodníku.

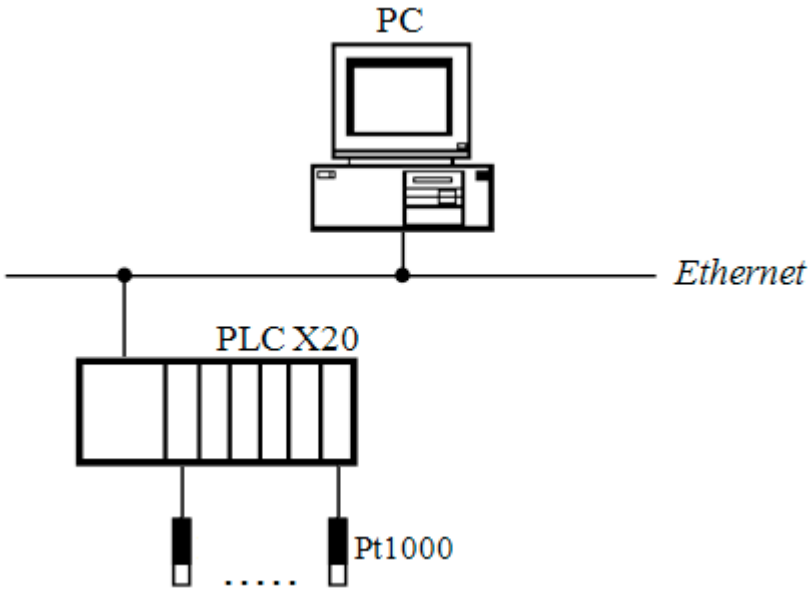
Jednou z hlavních kapitol je pátá kapitola. Pátá kapitola popisuje vývoj testovací aplikace určené pro odzkoušení činnosti převodníku a mikroLAN sítě. Šestá kapitola se věnuje vývoji realizované knihovny. Protože je potřeba nějakým způsobem realizovanou knihovnu odzkoušet věnuje se sedmá kapitola implementaci knihovny do reálného systému a testovacího systému. Reálným systémem bude zdokonalený systém měření teplot tepelných vrtů. Součástí této kapitoly je i vývoj vizualizační aplikace v Promoticu.

2 Rozbor aplikace pro měření teploty

Celá aplikace pro měření teploty je založena na měření teplot v tepelném vrtu. Čili součástí aplikace je několik tepelných vrtů a centrála s měřicím zařízením umístěná na budově VEC2 (Výzkumné energetické centrum 2). Jako měřící zařízení slouží programovatelný automat B&R Systém X20, který zpracovává data od jednotlivých čidel umístěných ve vrtech. Dále soustava aplikace ve sávajícím stavu obsahuje tepelné čerpadlo, kabeláž a teplotní čidla Pt1000.

2.1 Stávající stav systému

Obrázek 1 zobrazuje schéma stávajícího měřícího systému. Základem je programovatelný automat řady X20. X20 pomocí přídavných modulů zpracovává data od odporových teplotních čidel Pt1000 umístěných v jednotlivých vrtech (viz. Obr. 2). Jedná se o čidla s proměnným odporovým výstupem (velikost odporu odpovídá určité teplotě). Tyto čidla jsou připojeny na rozšiřující modul X20AT4222. Součástí stávajícího systému je i klasický osobní počítač, který komunikuje s PLC pomocí sběrnice Ethernet. Na tomto počítači je nainstalováno vývojové prostředí B&R Automation Studio pomocí něhož lze dále vyvíjet řídicí program PLC. Dále lze na tomto počítači pracovat s vizualizačním prostředí Promotic ve, kterém je vytvořena stávající aplikace vizualizace. Prostřednictvím této vizualizace je možno sledovat aktuální teplotu na jednotlivých čidlech systému. [8]



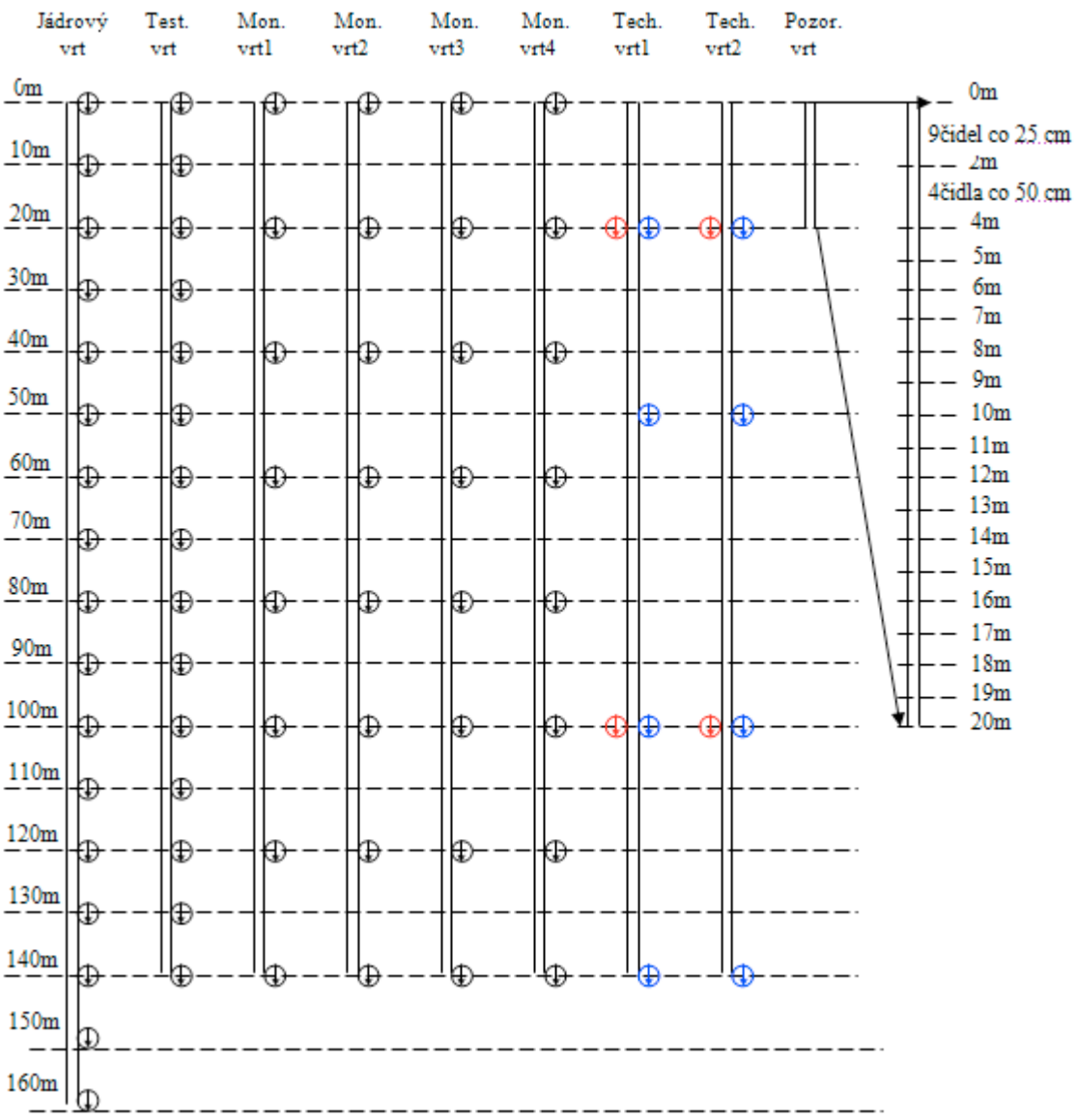
Obr.1: Hierarchické schéma stávajícího systému [8]

Každý složitější řídicí systém je logicky rozčleněn do několika vrstev, které se liší rozdílnou oblastí řízení. Na nejnižší vrstvě (vrstva přímého řízení) se nachází odporová teplotní čidla Pt1000 připojená k PLC přes rozšiřující moduly. PLC X20 komunikuje s nadřazenou vrstvou (HMI/SCADA) prostřednictvím komunikační sběrnice Ethernet. V této vrstvě se nachází již zmiňovaný osobní počítač s příslušným vizualizačním prostředím. [1], [8]

2.2 Hlubinné vrty

Stávající systém obsahuje 9 hlubinných vrtů, které jsou rozmístěny kolem budovy VEC2 (Výzkumné energetické centrum 2). Rozvržení hlubinných vrtů je do spirály, aby každý hlubinný vrt měl jinou vzdálenost od měřící soustavy a tím se daly pozorovat ztráty na vedení. Jednotlivé charakteristiky hlubinných vrtů [8]:

- Pozorovací vrt – Jednotlivé čidla jsou rozmístěna blízko sebe, slouží k detailnímu snímání povrchových teplot. Do 2m v rozmezí 25cm, od 2m v rozmezí 50cm a pak co metr jedno čidlo až do hloubky 20m.
- Technologický vrt – Systém obsahuje dva vrty. Slouží k ohřevu přiváděné studené vody pomocí speciálního vedení, ve kterém proudí voda do tepelného čerpadla. Celková hloubka vrtu je 140m. Tyto vrty mají dva druhy čidel, jedny slouží ke snímání studené vody a druhé ke snímání ohřáté vody. Rozmístění těchto snímačů je na Obr.2.
- Monitorovací vrt – Systém obsahuje čtyři vrty, slouží k monitorování teplot do hloubky 140m v rozmezí 20m.
- Testovací vrt – Tento vrt je vyhlouben do stejně hloubky jako monitorovací vrt, ale rozvržení čidel je dvojnásobné. Tedy co 10m.
- Jádrový vrt – U tohoto vrtu se měří teploty až do hloubky 160m, přičemž čidla jsou rozmístěna co 10m.

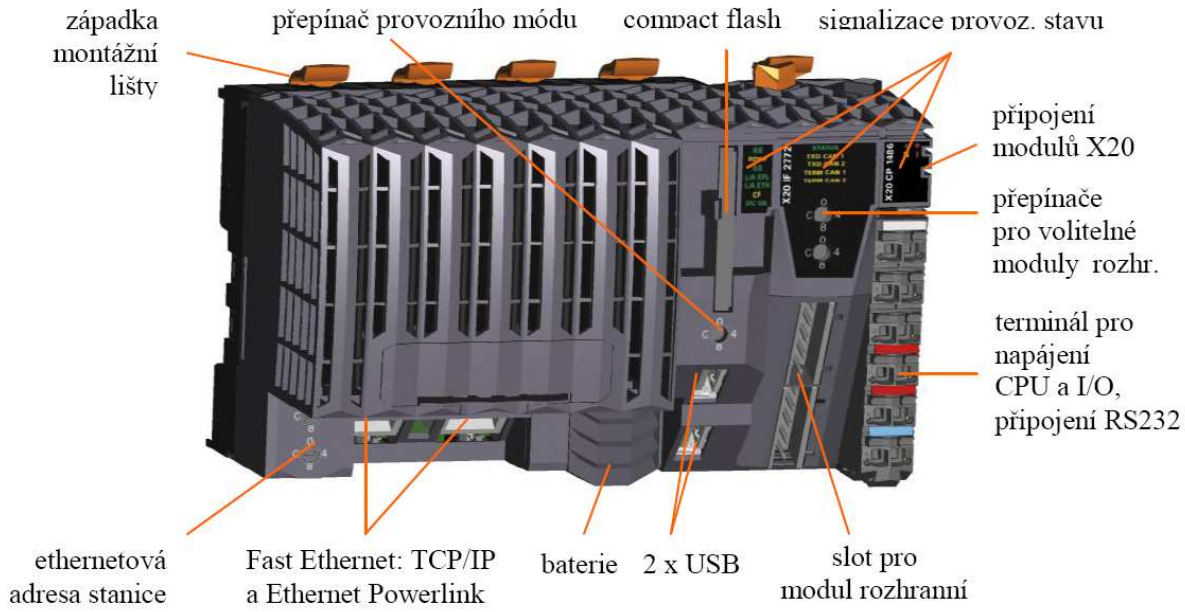


Obr.2: Hlubinné vrty – Rozmístění čidel [8]

V našem případě se jako měřící vrt používá tzv. Pozorovací vrt ve, kterém jsou umístěny požadované teplotní čidla s rozhraním 1-Wire. Těmito čidly jsou čidla DS18B20 od firmy Dallas Semiconductor (Maxim). [8]

2.3 PLC B&R Systém X20

Jedná se o sendvičový systém modulů vstup/výstup, které mohou být použity buď jako součást systému distribuovaných vstupů a výstupů nebo ve formě kompaktních modulárních řídicích systémů.



Obr.3: Procesorová jednotka Systému X20 [8]

Na Obr. 3 je znázorněna procesorová jednotka X20 CPU. K dispozici je v několika variantách, jejichž technické parametry jsou odstupňovány tak, aby vyhověly pro běžné použití i pro rozsáhlé a náročné aplikace. V podstatě jde o průmyslové počítače určené k montáži na lištu DIN. Obsahují standardní počítačový procesor, matematický koprocesor, paměť RAM a další komponenty běžných PC. U výrobků společnosti B&R patří ke standardnímu příslušenství rozhraní pro Ethernet (10/100 Mb/s, konektor RJ45) a dvě rozhraní USB. Mimo to mají X20 CPU v základní sestavě ještě rozhraní pro Ethernet Powerlink a jejich komunikační možnosti doplňuje rozhraní RS232. [2]

Nejkratší doba cyklu u X20 CPU může být až 200 mikrosekund, tzn. že je možné využít je i pro velmi přesné řízení rychlých dějů. Paměť RAM více než dostačuje pro většinu aplikací. Je navíc doplněna pamětí SRAM, jejíž napájení je zálohováno baterií. Do paměti SRAM lze uložit např. hodnoty parametrů specifických pro danou úlohu nebo remanentních proměnných. Hodnoty remanentních proměnných jsou po odpojení napájení automaticky překopírovány z RAM, jejíž obsah se bez napájení ztratí, do paměti SRAM. Při opětovném spuštění CPU jsou hodnoty z SRAM přesunuty zpět do RAM. [2]

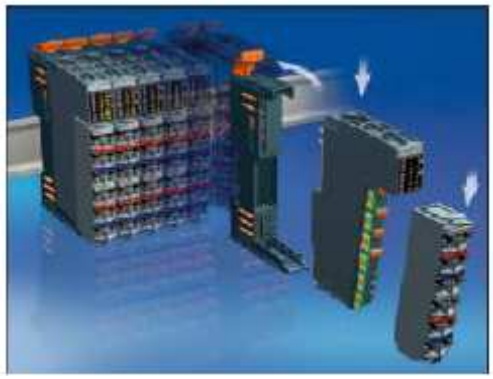
Pro zálohování dat, aplikačních programů, receptur nebo nastavení lze využít kartu Compact Flash. Další možností, jak zálohovat data, je připojit paměťové médium prostřednictvím USB.

Použité procesory Intel Celeron mohou pracovat i bez chlazení větrákem. Je-li však větrák použit (u procesoru Celeron 650), rozšiřuje se rozsah provozních teplot. Sledování teploty procesoru a činnosti větráku umožňuje zasáhnout při poruše a předejít tak zničení procesoru. Větrák lze vyměnit i za provozu zařízení. [2]

Kromě již uvedených vestavěných rozhraní USB, Ethernet a Ethernet Powerlink může být X20 CPU doplněn dalším jedním až třemi komunikačními moduly. K dispozici je několik typů komunikačních modulů: např. komunikační modul pro Ethernet Powerlink rozšiřuje základní sestavu o

dvě další rozhraní pro tuto sběrnici; v komunikačním modulu je navíc vestavěn rozbočovač (HUB). Další variantou je komunikační modul s rozhraním CAN, popřípadě v kombinaci s X2X Link. Dále jsou nebo v krátké době budou k dispozici moduly pro RS232 a Profibus-DP (Master nebo Slave). [2]

Vstupní a výstupní moduly systému X20 lze jednoduše začlenit i do rozsáhlých decentralizovaných automatizačních systémů, které používají průmyslové sběrnice Profibus-DP, CANopen, DeviceNet, či ETHERNET Powerlink. Moduly systému X20 se vyznačují velmi účelným mechanickým provedením, které umožňuje nejenom snazší montáž, ale také například výměnu jednotlivých modulů pod napětím, nebo efektivnější logistiku. [2]



Obr.4: I/O přídavné moduly Systému X20 [2]

K jednomu X20 CPU lze připojit až 250 vstupních/výstupních modulů. To představuje až 3 000 vstupních a výstupních kanálů. Jsou dvě možnosti, jak vstupní/výstupní moduly systému X20 připojit k X20 CPU: buď je mechanicky i elektricky spojit navzájem do jednoho celku (montují se na pravou stranu CPU), nebo nosnou desku (backplane) dále rozčlenit. Možnost rozčlenění je základní vlastností nosné desky, která nese označení X2X Link. Označení X2X Link je hlavně označení pro samotnou komunikační sběrnici, přes kterou dané přídavné moduly komunikují. V obou případech základní deska zajišťuje jak přenos signálů, tak napájení, přičemž zvnějšku z pohledu ovládání a komunikace není podstatné, zda je modul připojen k CPU přímo nebo prostřednictvím kabelu – jde o tzv. distribuovanou základní desku (remote backplane). [2]

2.3.1 X20CP1484

Jedná se o modul kompaktního CPU systému X20. Základem CPU je procesorová technologie Intel Celeron. Jedná se o nejnižší model kompaktního CPU z nabídky firmy B&R systému X20. Tohoto řešení se velice často využívá především z pohledu velice jednoduchého a rychlého řešení jakýchkoliv úprav, které se obejde s minimem finančních nákladů. Provozní rozsah teplot je u těchto kompaktních CPU daleko menší než v případě aktivně chlazených. Jeho základní technické údaje jsou [2]:

- Procesor Intel Celeron 266 MHz.
- 32 MB paměti DRAM, 1 MB paměti SRAM, aplikační paměť Compact Flash.
- 1 slot pro moduly X20IF.
- 2 USB rozhraní a 1 RS232 rozhraní.

- 1 rozhraní Ethernet 10/100 Base-T, 1 ETHERNET Powerlink (podpora profilu ETHERNET Powerlink).
- Čas nejrychlejšího tasku – 800 µs.

Obr.5: X20CP1484 [2]

Stavy LED diod:

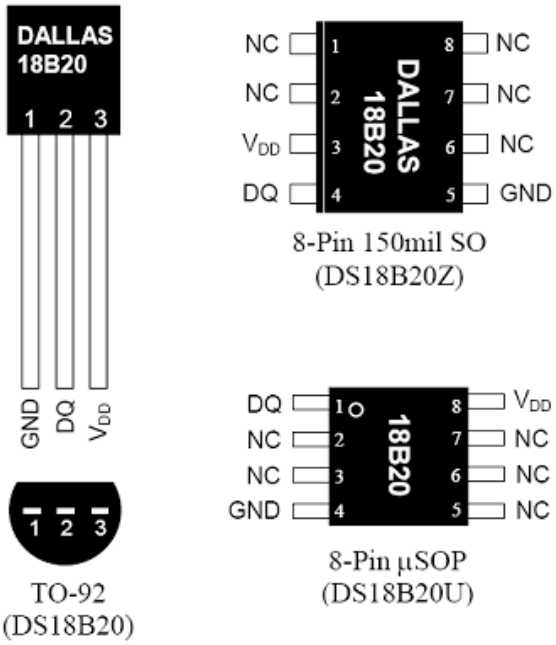
R/E	Zelená (svítí)	Aplikace je spuštěna.
	Červená (svítí)	Servisní mód.
RDY/F	Žlutá (svítí)	CPU je aktivní.
	Zelená (svítí)	Překročení teploty.
S/E	Zelená/Červená	Status „Chyba“.
EPL	Zelená (svítí)	Bylo navázáno spojení s Powerlink vzdálenou stanicí.
	Zelená (bliká)	Bylo navázáno spojení s Powerlink vzdálenou stanicí, LED dioda bliká, jestliže je zjištěna aktivita na sběrnici Ethernet.
ETH	Zelená (svítí)	Bylo navázáno spojení s Ethernet vzdálenou stanicí.
	Zelená (bliká)	Bylo navázáno spojení s Ethernet vzdálenou stanicí, LED dioda bliká, jestliže je zjištěna aktivita na sběrnici Ethernet.
CF	Žlutá (svítí)	Paměť CompactFlash je v pořádku.
	Zelená (svítí)	Paměť CompactFlash je aktivní.
DC OK	Zelená (svítí)	Napájení CPU je v pořádku.
	Červená (svítí)	Zálohovací baterie je vybitá.

Tab.1: Stavy X20CP1484 [8]

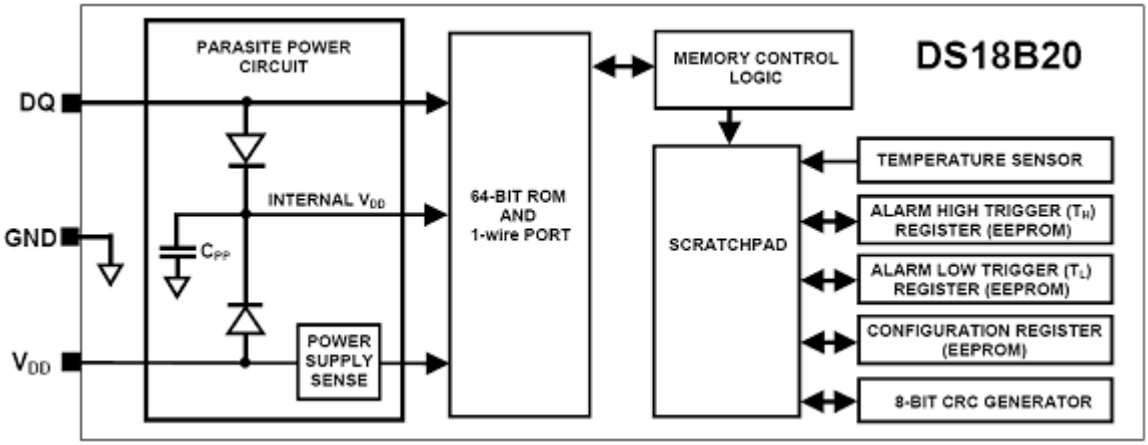
2.4 Čidla DS18B20

Jedná se o digitální teploměr s rozhraním 1-Wire. Umožňuje měřit teplotu v rozsahu od -55°C až do +125°C s přesností ±0,5°C garantovanou v rozsahu od -10°C až do +85°C. Samotný rozsah dat teplotního čidla je možno měnit od 9 do 12 bitů (9 bitů odpovídá přesnosti 0,5°C, 10 bitů odpovídá přesnosti 0,25°C, 11 bitů odpovídá přesnosti 0,125°C a 12 bitů odpovídá přesnosti 0,0625°C) pomocí příslušného konfiguračního registru (Configuration registr). Ve standardním nastavení je rozsah 12 bitů. Převod teploty na digitální slovo trvá 750 milisekund (maximálně). Změřená teplota se ukládá do 2-byte registru (Temperatur sensor). Pokud je doba převodu delší nebo dojde k jiné komplikaci vygeneruje se chybová hláška v alarmovém registru (alarm registr – Alarm TH a TL). Každé čidlo má své unikátní 64 bitové sériové číslo, uchovávané v ROM paměti, které slouží k adresaci konkrétního senzoru. Toto číslo usnadňuje rozpoznání čidla a následnou komunikaci. 64 bitové sériové číslo

společně s unikátním protokolem identifikace připojených zařízení ke sběrnici umožňuje provozovat několik senzorů na jednom datovém vodiči. Samotné čidlo obsahuje tři výstupy, kde dva slouží pro napájení (GND a VDD) a třetí (DQ) slouží pro komunikaci po 1-Wire sběrnici. Napájecí napětí se pohybuje v rozmezí od 3,3V až po 5,5V. Pro případ dvouvodičového (pouze GND a DQ) propojení se sítí 1-Wire obsahuje kondenzátor (Cpp), který slouží jako nábojová pumpa a dodává obvodu společně s datovým vodičem dostatečné napájecí napětí pro plnohodnotnou činnost. [3].



Obr.6: Čidlo DS18B20 [3]



Obr.7: Blokové schéma čidla DS18B20 [3]

3 MicroLAN

Jedná se o jednoduchou a levnou síťovou technologii pro automatizaci. Používá protokol 1-Wire, tudíž i sběrnici 1-Wire. Její komponenty jsou přístroje a zařízení s 1-Wire rozhraním a blokem 64 bitového identifikačního čísla (8 bitů CRC, 48 bitů sériového čísla a 8 bitů typu zařízení). Identifikační číslo umožňuje jednoznačnou identifikaci zařízení na síti. Zařízení obsahující tyto dvě části je výrobcem deklarováno, jako standard MicroLAN (Standard IEEE 1451.4). Dalšími součástmi sítě mohou být různé adaptéry a kontroléry. [5], [6]

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (28h)	
MSB	LSB	MSB	LSB	MSB	LSB

Obr.8: Identifikační číslo – ROM kód (28h je family code čidla DS18B20) [3]

3.1 1-Wire

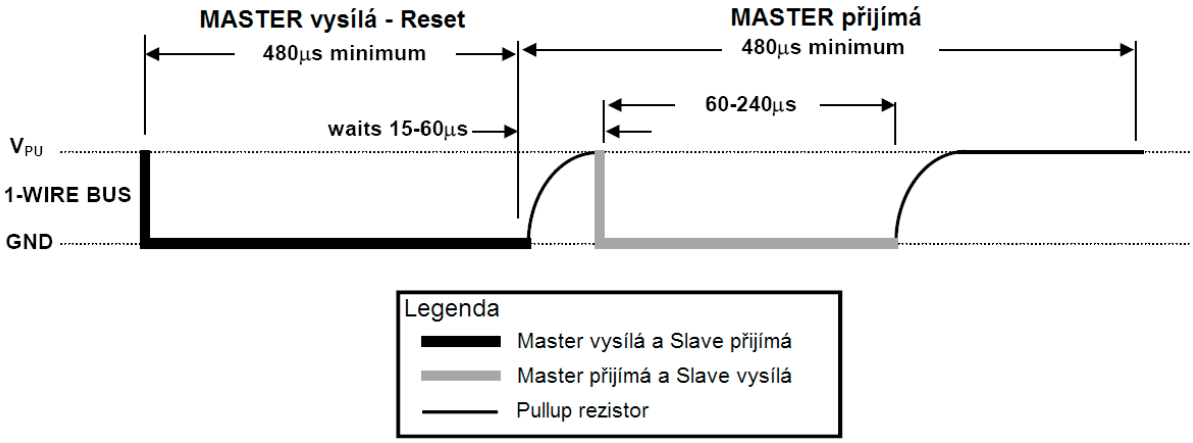
1-Wire je založen na komunikaci mezi jedním zařízením typu Master a několika zařízeními typu Slave. Jednotlivé zařízení jsou propojeny paralelně pomocí jednoho datového vodiče a dvou napájecích vodičů. Mezi datový vodič a napájecí vodič (VDD) se umisťuje tzv. Pullup rezistor (asi 5kΩ), který uvede datový vodič do logické 1. Mimo to sběrnice 1-Wire umožňuje i napájení Slave zařízení po datovém vodiči. Pro tento účel napájení musí zařízení typu Slave obsahovat kondenzátory plnící funkci nábojové pumpy. 1-Wire se používá pro standard TTL/CMOS v rozsahu 2,8V až 6V. [6]

Síť založená na 1-Wire sběrnici je velice jednoduchá a je určena převážně pro připojení různých typů čidel snímající určitou fyzikální veličinu (teplota, tlak, poloha apod.). Zařízením typu Slave je většinou digitální čidlo. Toto čidlo může být přímo s podporou sběrnice 1-Wire nebo může obsahovat převodník, na který se připojí konkrétní čidlo snímající určitou fyzikální veličinu. Zařízení typu Master musí být takové zařízení, které dovoluje vysílat a přijímat data a příkazy po sběrnici 1-Wire. Takovým zařízením je většinou mikrokontrolér nebo převodník převádějící komunikaci 1-Wire na jinou (sériová komunikace – RS232, USB apod.). [5], [6]

Komunikace je asynchronní a poloduplexní. Data se po sběrnici přenášejí v takzvaných časových slotech. Při komunikaci se 1-Wire komponenty samy taktují vyhodnocením délky časových intervalů impulzních signálu. Pokud Master zařízení vysílá, přijímají všechny zařízení typu Slave, ale odpovídá pouze 1 Slave. Komunikační rychlost 1-Wire se pohybuje od 16kbit/s (velké vzdálenosti – maximálně 750m) do 125kbit/s (krátké vzdálenosti). [6]

3.1.1 Inicializace - Reset impulz a přítomnostní pulzy

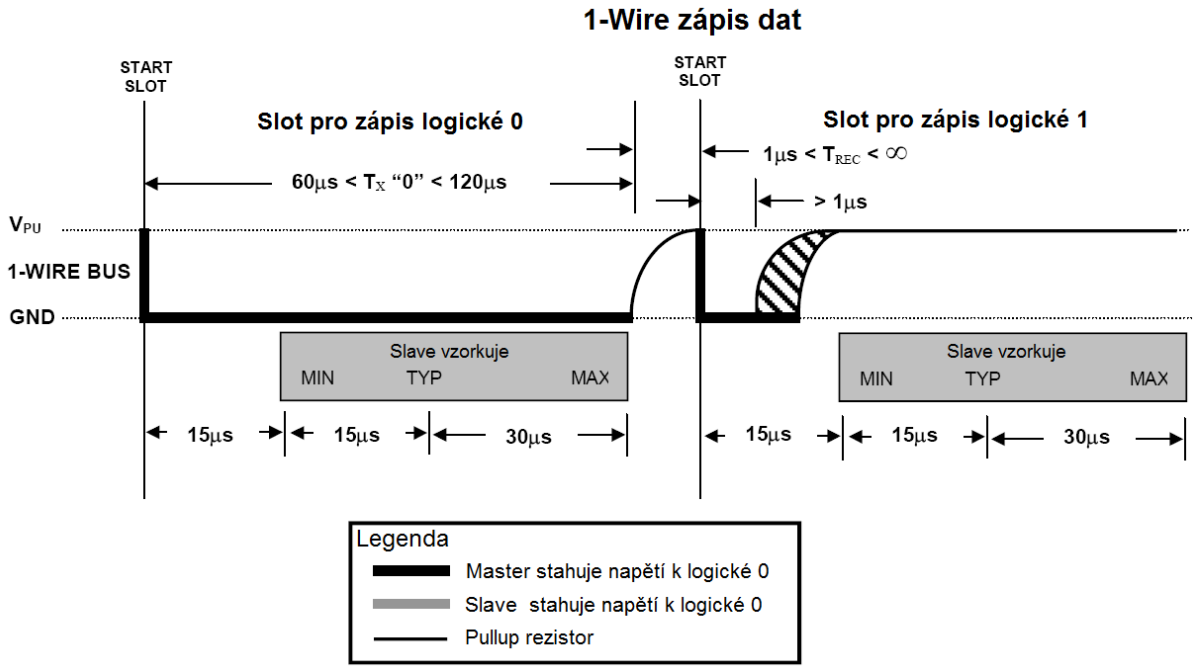
Komunikaci zahajuje vždy zařízení typu Master reset pulzem (Inicializační sekvence). Master nejprve stáhne datový vodič do logické 0 (uzemní se) a drží ho na této úrovni minimálně 480 mikrosekund. Pak sběrnici uvolní a naslouchá. Pullup rezistor zatím vrátí sběrnici zpět do logické 1. Pokud je na sběrnici připojené nějaké 1-Wire zařízení, tak detekuje tuto vzestupnou hranu a po prodlevě 15 až 60 mikrosekund stáhne sběrnici na 60 až 240 mikrosekund k logické 0. [3], [7]



Obr.9: Inicializační sekvence [3]

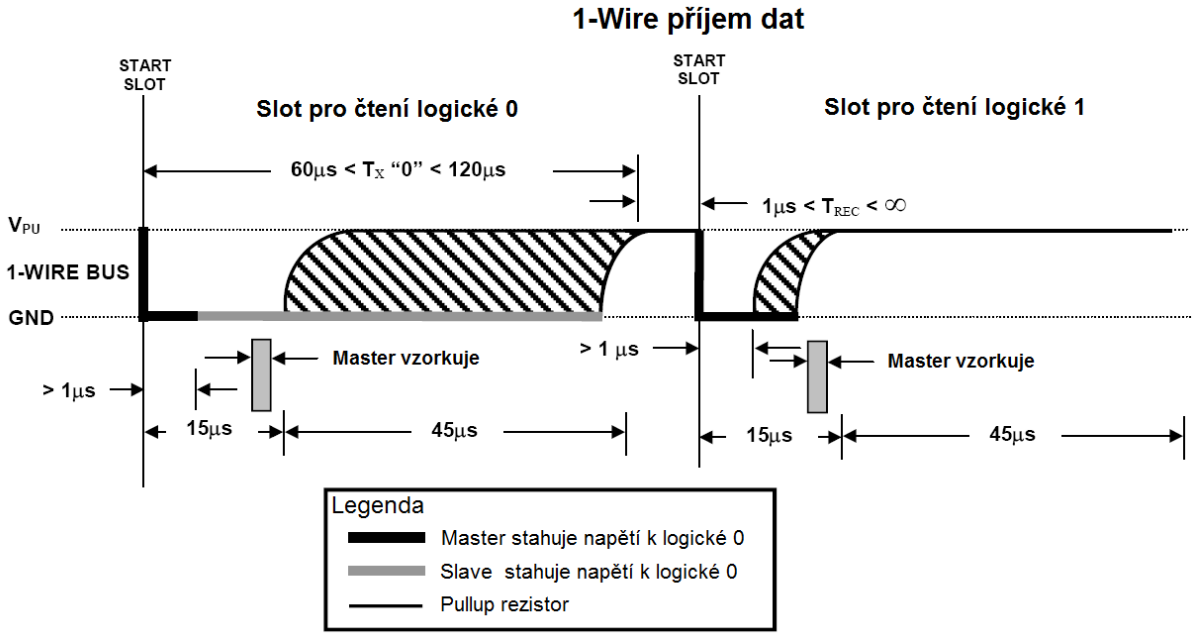
3.1.2 Vysílání a příjem dat

Pokud se zařízení Slave správně ohlásí, může Master začít vysílat a přijímat data. Data jsou vysílána v tzv. časových slotech (časové okénka). Slot je dlouhý 60 až 120 mikrosekund a během jednoho slotu je vyslán nebo přijat jeden bit informace. Mezi jednotlivými sloty musí být minimálně 1 mikrosekunda mezera, kdy je sběrnice v klidu. Existují 4 druhy časových slotů (zápis logické 1, zápis logické 0, čtení logické 1 a čtení logické 0). Zápisové sloty slouží k tomu, aby Master vyslal data nebo příkazy do zařízení typu Slave. Zápis logické 1 probíhá tak, že Master stáhne sběrnici k logické 0 minimálně na 1 mikrosekundu a nejpozději do 15 mikrosekund od začátku ji opět uvolní a ponechá uvolněnou. Pullup rezistor ji tedy vytáhne k logické 1. Zápis logické 0 je o něco jednodušší. Master stáhne sběrnici k logické 0 a ponechá ji tak po celý slot, tedy minimálně 60 mikrosekund. Zařízení vzorkuje stav na datovém vodiči zhruba 30 mikrosekund po začátku časového slotu. [3], [7]



Obr.10: 1-Wire zápis dat [3]

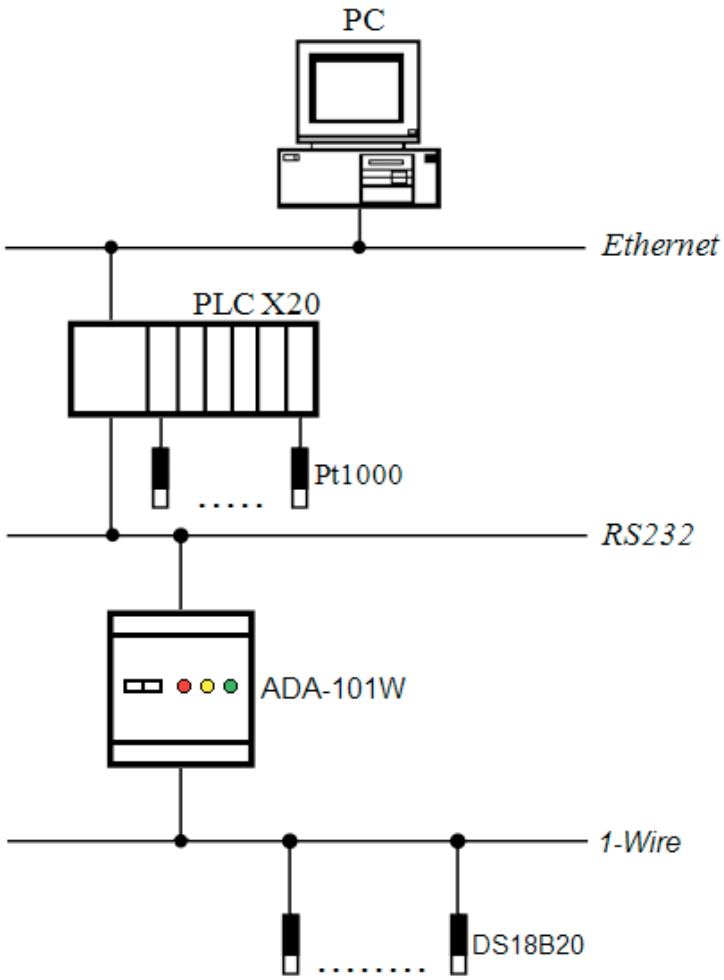
Čtecí sloty opět inicializuje Master tím, že stáhne sběrnici k logické 0 na minimálně 1 mikrosekundu a opět ji uvolní. Po tomto zahájení může zařízení vyslat 1 bit buď tím, že ponechá sběrnici v klidu (logická 1) nebo že ji stáhne (logická 0). [7]



Obr.11: 1-Wire příjem dat [3]

4 Návrh systému s 1-Wire

Obrázek 12 zobrazuje schéma upraveného měřícího systému. Základem systému zůstává programovatelný automat řady X20. Stejně tak zůstávají součástí systému rozšiřující moduly s teplotními čidly Pt1000 a propojení systému s klasickým počítačem přes Ethernet.



Obr.12: Hierarchické schéma systému s 1-Wire

Novinkou v systému je mikroLAN na niž jsou připojeny digitální teplotní čidla DS18B20. Síť mikroLAN je realizována na základě rozhraní 1-Wire. Pro realizaci sběrnice 1-Wire je využit převodník ADA-101W, který převádí sériovou komunikaci sběrnice RS232 na protokol 1-Wire microLAN sítě. Převodník ADA-101W je napojen na rozhraní RS232 programovatelného automatu X20 a zároveň na sadu teplotních čidel DS18B20 umístěných v pozorovacím vrtu (viz. Obrázek 2). Z toho vyplývá, že data z 1-Wire sběrnice přímo zpracovává procesorová jednotka X20CP1484 systému X20.

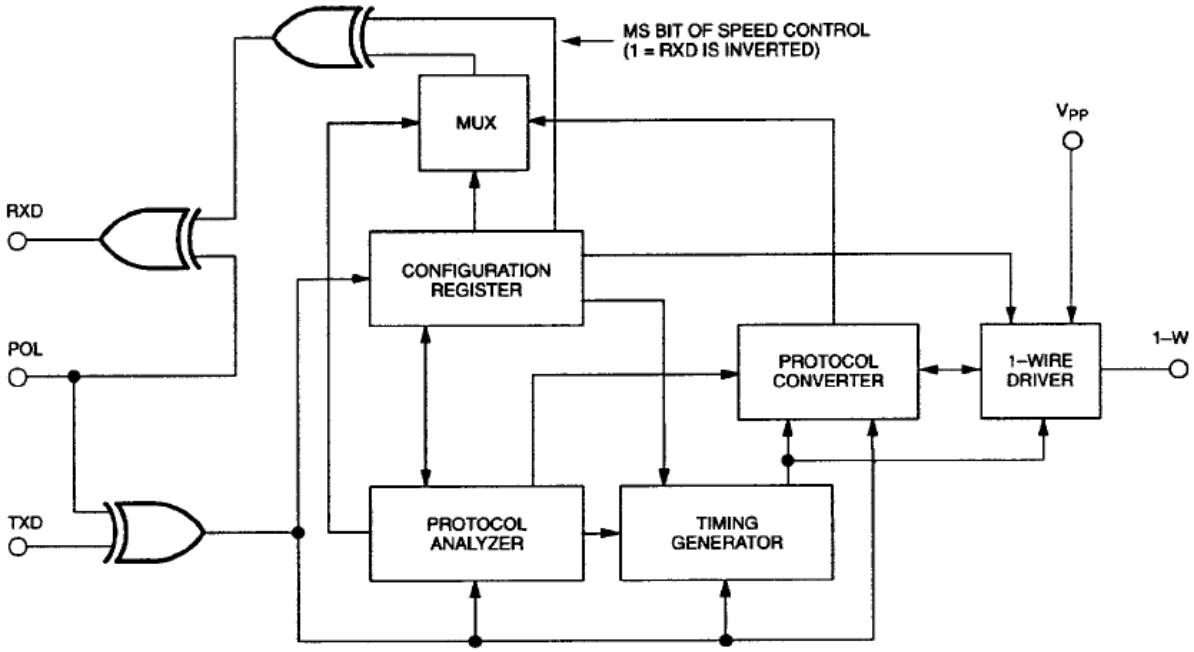
4.1 Převodník ADA-101W

Jedná se o průmyslový převodník s převodem rozhraní RS232 na 1-Wire rozhraní a naopak. Umožňuje připojení teplotních čidel, obvodů reálného času, EEPROM pamětí apod. Tento převodník

je schopen zpracovávat signály z 1-Wire rozhraní o přenosové rychlosti ve standardním režimu až 16,3 kbit/s, maximální rychlost až 142kbit/s. Převodník je typu 2-2-2, což znamená, že poskytuje možnost programovat 1-Wire rozhraní. Komunikace mezí jednotlivými sítěmi je signalizována žlutou a červenou LED (Rx a Tx). Používá napájení 10 až 30V s minimálním výkonem 2W. Využívá galvanického oddělení mezi sériovým rozhraním a napájecím zdrojem. Taktéž je galvanicky odděleno rozhraní RS232 a 1-Wire rozhraní. Hlavní výhodou převodníku je, že obsahuje klasické rozhraní RS232, které je kompatibilní s rozhraním RS232 klasického počítače díky němuž je možné odzkoušet 1-Wire síť i na klasickém počítači. Provedení převodníku odpovídá normě IP40. Je určen pro umístění na DIN lištu stejně jako PLC X20 což umožňuje zkrácení kabeláže mezi PLC a převodníkem na minimum. [4]

Obr.13: Převodník ADA-101W [4]

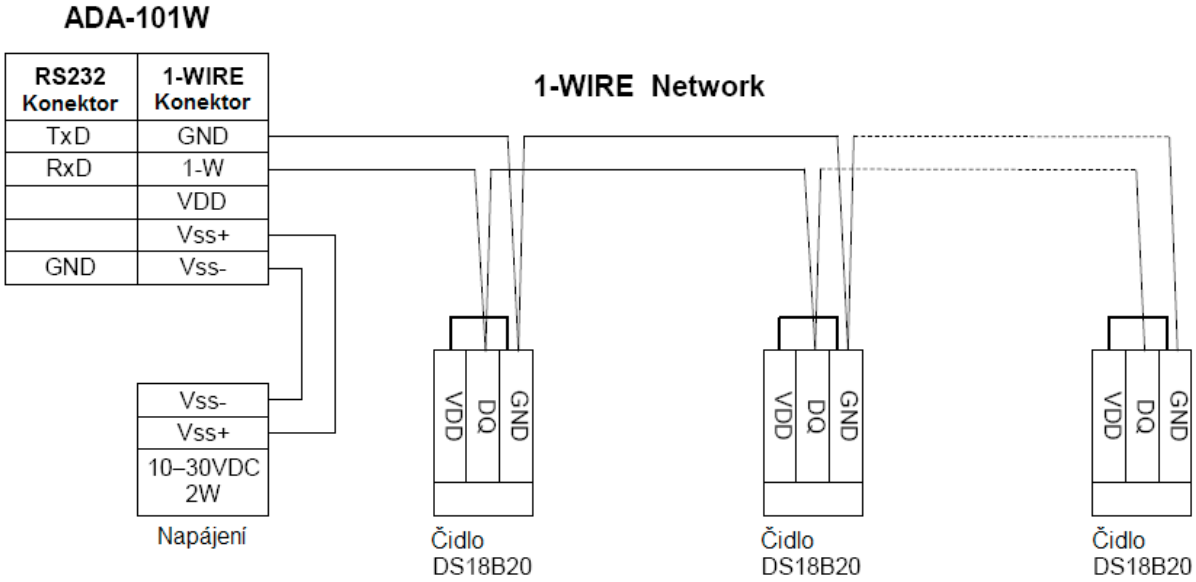
Hlavní částí, která provádí převod komunikace RS232 na 1-Wire komunikaci je převodník DS2480B. Jedná se o převodník, který má již integrován základní protokol 1-Wire (Protocol Analyzer a Protocol Converter) a na základě přijatých dat ze sériové linky generuje časové posloupnosti (Timer Generator) reprezentující komunikaci po 1-Wire lince. Díky tomuto převodníku má převodník ADA-101W podobnou funkci jako převodník DS9097U. [4]



Obr.14: Blokové schéma DS2480B [9]

4.2 Topologie 1–Wire sítě

1-Wire síť má podobu kořenové topologie. Zařízením Master je v této síti převodník ADA-101W a Slave zařízením jsou teplotní čidla DS18B20. Samotná sběrnice je tvořena jednoduchým dvouvodičovým (zemnící vodič GND a datový vodič DQ) vedením z kroucené dvojlinky, která začíná na vývodech Master zařízení (pin označené DQ a GND) a končí na vývodech nejvzdálenějšího Slave zařízení. Ostatní Slave zařízení jsou na sběrnici připojeny ve formě větví z čehož vyplývá, že Slave zařízení jsou k Master zařízení připojeny paralelně. Kroucená dvojlinka by měla být minimálně typu STP 1x2x0,5(24AWG). V případě dvouvodičového propojení je třeba u jednotlivých Slave zařízení propojit pin VDD s pinem GND.



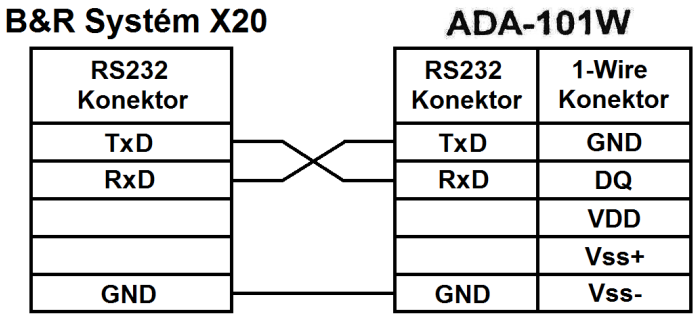
Obr.15: Dvouvodičová topologie sítě 1-Wire [4]

Dvou linková topologie má v tomto případě určitou nevýhodu a zároveň i výhodu. Touto nevýhodou je napájecí napětí, které je odvozeno z datové linky (parazitní napájení) 1-Wire (DQ) což znamená, že na větší vzdálenosti dochází k většímu útlumu než v případě tří linková topologie. Výhodou je její jednoduchost a zároveň dochází k menšímu namáhání kabeláže přímo ve vrtu. Pro případ měření teplot v pozorovacím vrtu je dvouvodičová topologie dostačující. Pro případ měření teplot například v jádrovém vrtu by se měla použít tří linková topologie z důvodu menšího útlumu. V případě tří linkových vedení se nepropojují piny VDD s GND jednotlivých Slave zařízení, ale propojí se jednotlivé piny VDD Slave zařízení s vývodem VDD Master zařízení (paralelní propojení).

4.3 Topologie RS232

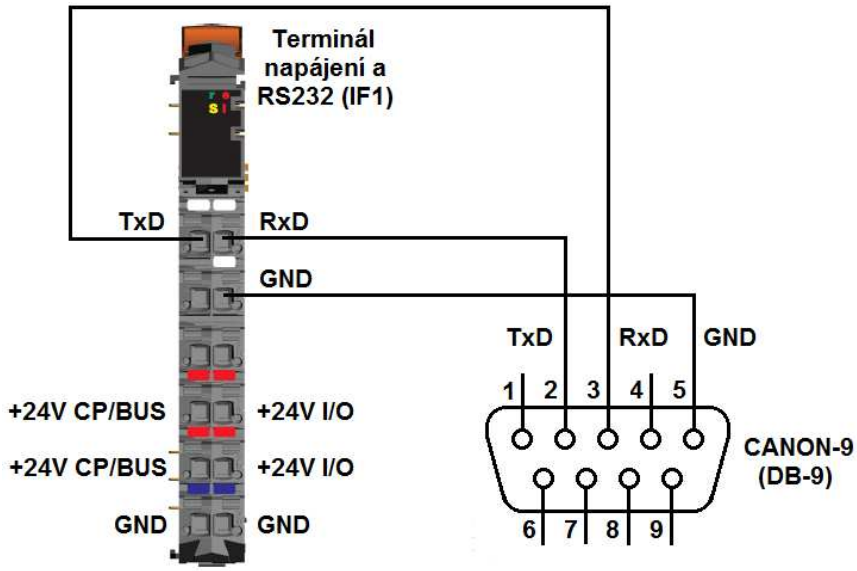
RS232 síť má stejnou topologii jako 1-Wire síť. Tato síť obsahuje pouze jedno zařízení typu Master a jedno zařízení Slave. Zařízením Master je v tomto případě PLC systém X20 a Slave zařízením je převodník ADA-101W. Samotná sběrnice je tvořena tří linkovým vedením (zemnící vodič GND, vysílací vodič TxD a přijímací vodič RxD). Aby fungovala komunikace po RS232 v duplexním režimu jsou propojeny vždy vývody vysílače (TxD Master – PLC systém X20)

s přijímačem (RxD Slave - převodník ADA-101W) a přijímače (RxD Master – PLC systém X20) s vysílačem (TxD Slave - převodník ADA-101W), což je vidět na obrázku 16 a 17.



Obr.16: Topologie sítě RS232

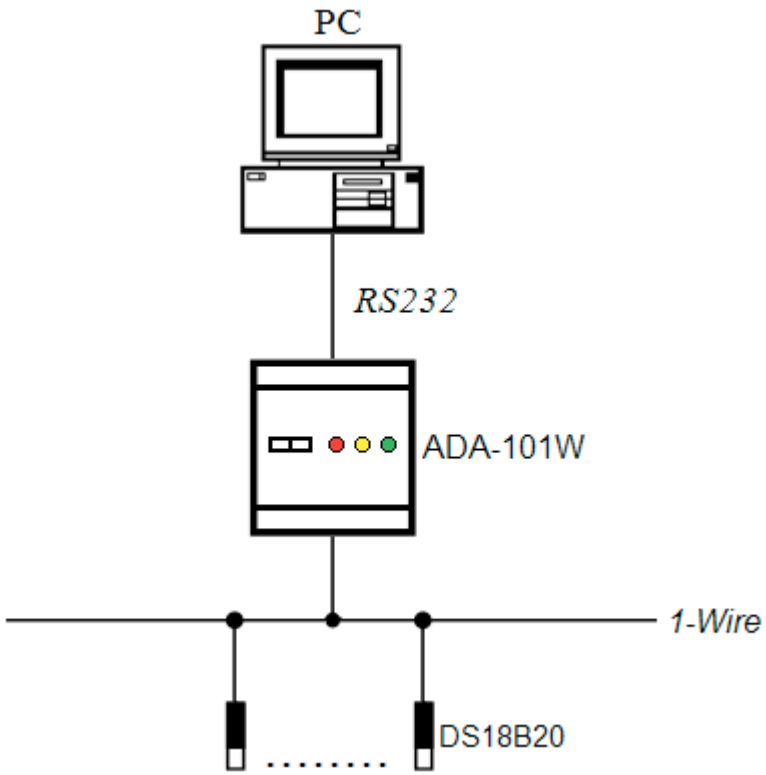
Při reálném propojení Master zařízení a Slave zařízení se jeden konec tří linkového vedení připojí na terminál IF1 (viz. Obrázek 3 a 17) do montážních otvorů pro RS232 komunikaci a druhý konec se připojí přes konektor Canon-9 (DB-9 Samec) ke konektoru Canon-9 (DB-9 Samice) převodníku ADA-101W.



Obr.17: Reálné zapojení RS232

5 Návrh testovacího programu pro počítač

Testovací program pro počítač slouží jako demonstrační úloha měření teploty a vyhledávání 1-Wire zařízení (DS18B20). Přínosem tohoto programu je z hlediska návrhu knihovny pro PLC systém X20. Základní konstrukční prvky tohoto programu se implementují do knihovny PLC. Aby bylo možné použít testovací program je potřeba propojit počítač a převodník s mikroLAN sítí (viz. Obrázek 18). Počítač musí obsahovat port RS232.



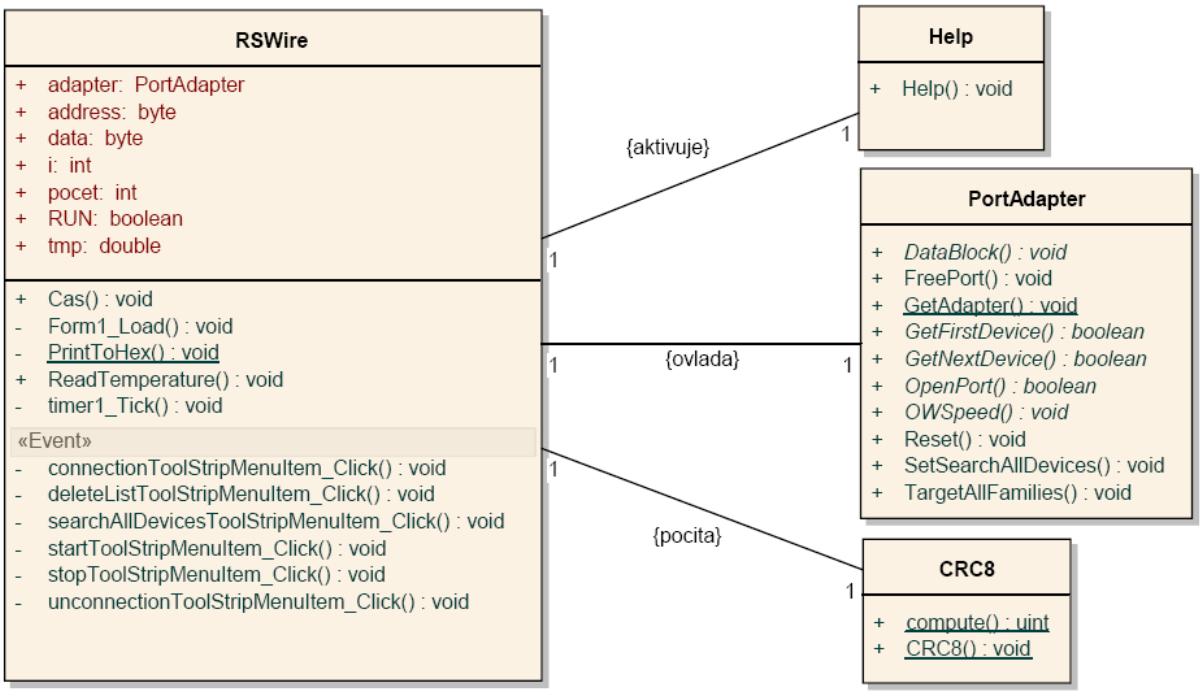
Obr.18: Hierarchické schéma propojení počítače a mikroLAN sítě

Testovací program je vytvořen ve vývojovém prostředí Microsoft Visual Studio pomocí jazyka C#. Samotný program se tedy skládá z vizuální části a části kódu. Vizuální část je část, která tvoří okno programu při jeho aktivaci.

5.1 Realizace Programu

Pro realizaci programu se vychází z parametrů požitých zařízení (DS18B20 a DS2480B) a ze základních metod dostupných na stránkách výrobce 1-Wire zařízení (Dallas Semiconductor – Maxim). Jedná se o metody realizující sériový port (DalSemi.OneWire - DalSemi.Serial) některé převodové funkce (DalSemi.OneWire.Adapter) a CRC kód (DalSemi.Utils). Metody realizující sériový port jsou také součástí systémových knihoven (System.IO.Ports) vývojového systému. Pro zjednodušení práce je využita knihovna výrobce 1-Wire zařízení, kde je přímo nastaven název portu, název převodníku (DS2480B) a názvy jednotlivých datových rychlostí. V případě standardní systémové knihovny by se tyto parametry musely nastavovat. Nastavením datových rychlostí a zasláním konfiguračního bytu se

realizuje časování mikroLAN sítě (viz. Obrázek 14). Čili obvod DS2480B na základě datové rychlosti sériové linky a konfiguračního bytu generuje časy potřebné pro realizaci protokolu 1-Wire. Časováním se nastavuje datová rychlost 1-Wire. Metoda pro odeslání konfiguračního bytu je součástí převodové funkce (DalSemi.OneWire.Adapter) a vyvolává se metodou OWSpeed.



Obr.19: Diagram tříd testovacího programu

	Kód hodnoty								
Kód parametru	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**Jednotka**
001 (PDSRC)	15	2.2	1.65	1.37	1.1	0.83	0.7	0.55	V/µs
010 (PPD)	32	64	128	256	512	1024	2048	oo	µs
011 (SPUD)	16.4	65.5	131	262	524	1048	"dyn."	oo	ms
100 (W1LT)	8	9	10	11	12	13	14	15	µs
101 (DSO/W0RT)	3	4	5	6	7	8	9	10	µs
110 (LOAD)	1.8	2.1	2.4	2.7	3.0	3.3	3.6	3.9	mA
111 (RBR)	9.6	19.2	57.6	115.2	9.6	19.2	57.6	115.2	kbps

Tab.1: Hodnoty (bity) konfiguračního bytu [9]

PDSRC – PullDown Slew Rate Control – Rychlost PullDown (změna napětí za čas)
PPD – Programming Pulse Duration – Šířka programovacího pulzu
SPUD – Strong PullUp Duration – Silný PullUp (pro parazitní napájení 1-Wire)
W1LT – Write 1 Low Time – Zápis nízkých časů
W0RT – Write 0 Recovery Time – Zápis zotavovacích časů
LOAD – Load Sensor Threshold – Nastavení senzorového práhu
RBR – RS232 Baud Rate – Datová rychlost z RS232

Funkce	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(Zápis) Write parametru	0	Kód parametru			Kód hodnoty			1
(Čtení) Read parametru	0	0	0	0	Kód parametru			1

Tab.2: Konfigurační byte [9]

Co se týče struktury programu jsou na začátku programu definovány jednotlivé knihovny funkcí (systémové – System a přídavné – DalSemi). Za těmito definicemi následují jednotlivé funkce, které jsou volány na základě tzv. Eventu. Jedná se o změnu vyvolanou volbou konkrétní položky v menu (Options nebo Help).

Funkce timer1_Tick je funkce generována časovačem (Timer). Tato funkce se volá každou sekundu. Díky této funkci se cyklicky volá funkce realizující čas (DataTime time = DataTime.Now – hodnota time se pak zobrazuje) a funkce realizující měření teploty. Funkce měření se začne provádět, pokud byla aktivována proměnná RUN.

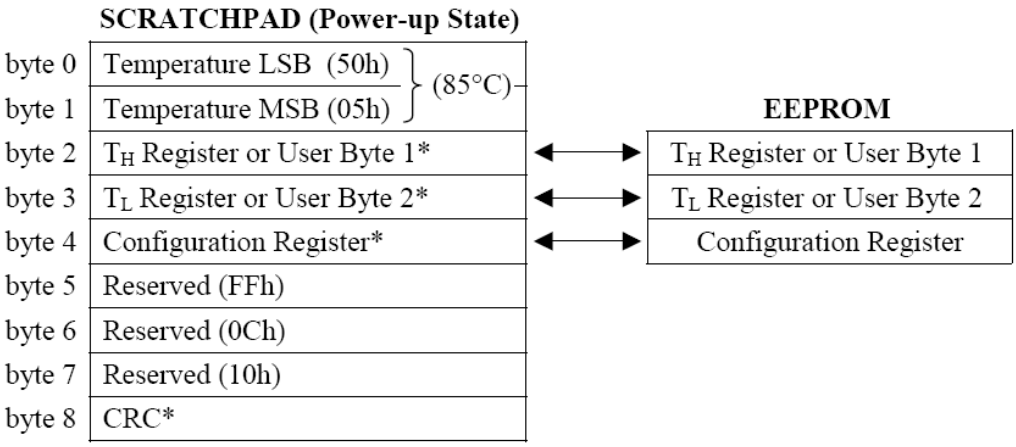
Dalšími základními funkcemi jsou funkce otevření a zavření sériového portu. Otevření portu se vyvolá volbou položky Connection čímž se vyvolá event příslušné funkce (connection - ToolStripMenuItem_Click). Na základě této funkce se volá metoda (adapter = AcssProvider.DefaultAdapter) pro otevření portu. Pokud nastane nějaká chyba nebo se v danném počítači nenachází sériová linka vyvolá se metoda Messagebox. Uzavření portu se vyvolá volbou položky Unconnection čímž se vyvolá event příslušné funkce (unconnectionTool - StripMenuItem_Click). Na základě této funkce se volá metoda (adapter.FreePort()) pro uzavření portu. Pokud nastane nějaká chyba vyvolá se metoda Messagebox. Metody pro otevření a uzavření portu se nachází v knihovně DalSemi a kromě několika nastavení jako je název portu a název adaptéru jsou tyto dvě metody ekvivalentní s metodami (Open() a Close()) v systémové knihovně IO.Ports.

Příkaz	Popis	Tvar (hex)
Convert T	Inicializace teploty	44h
Read Scratchpad	Čtení celého Scratchpad včetně CRC bytu	BEh
Write Scratchpad	Zápis dat do Scratchpad bytů 2,3 a 4	4Eh
Copy Scratchpad	Kopírování TH bytu, TL bytu a konfiguračního bytu do paměti EEPROM	48h
Recall E^2	Zpětné zavolání TH bytu, TL bytu a konfiguračního bytu z EEPROM do Scratchpadu	B8h
Read Power Supply	Signalizace napájecího režimu	B4h
Search ROM	Vyhledávání zařízení a identifikace ROM	F0h
Read ROM	Přečtení ROM v případě, že je na 1-Wire pouze jedno zařízení	33h
Match ROM	Zápis ROM příkazu, který posílá dotaz na konkrétní snímač	55h
Skip Match ROM	Příkaz pro přeskočení ROM. Umožňuje vyvolání teplotní konverze všech čidel na 1-Wire	CCh
Alarm Search	Signalizace alarmu	ECh

Tab.3: Tabulka příkazů snímače DS18B20 [3]

Vyvoláním eventu vyhledávání zařízení se zavolá příslušná funkce (searchAllDevicesTool - StripMenuItem_Click). V této funkci se jako první odešle resetovací puls a následně konfigurační

byte pomocí metody OWSpeed čímž se nastaví převodník DS2480B. Po tomto nastavení se cyklicky odešle byte s příkazem pro vyhledávání zařízení (F0h). Cyklicky se volá z toho důvodu, protože převodník na základě tohoto příkazu přečte ROM paměť 1-Wire zařízení (DS18B20) a v podobě 8 bytů jej odešle do počítače. Těchto 8 bytů se uloží do pole byte označené address. Po naplnění pole address se následně pole vyčte pomocí cyklu for a pomocí metody ToString("X2") se převede do hexadecimálního tvaru. Hexadecimální tvar se po převodu zobrazí do bloku 1-Wire (textBox).



Obr.20: Scratchpad a EEPROM snímače DS18B20 [3]

Funkce	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Zápis Resetu	1	1	0	X	00 - regular speed 01 - flexible speed 10 - overdrive speed 11 - regular speed		0	1
Odpověď na Reset	1	1	X	0	1	1	00 - 1-Wire krátký 01 - přítomnostní puls 10 - alarm puls 11 - žádný přítomnostní puls	

Tab.4: Reset byte [9]

Poslední funkcí je funkce pro měření teploty. Funkce se vyvolá na základě změny proměnné RUN. Začátek funkce je stejný jako u funkce vyhledávání. Provede se počáteční inicializace a po té se odešle příkaz pro vyhledání zařízení (F0h) s uložením přijatých bytů do pole byte označeném address. Stejně jako u vyhledávací funkce se i tato část provádí cyklicky dokola. Rozdílem je, že se přijaté byty adresy využívají ještě dále. Před dalším zpracováváním se odešle resetovací byte (pomocí metody Reset()). Po tomto příkazu se odešle Match ROM příkaz (55h) a s ním se odešle i jedna adresa z pole bytů address. Tím se vybere konkrétní snímač na mikroLAN síti. Nyní se odešle příkaz pro převod teploty (44h). Dojde k aktualizaci prvních dvou bytů Scratchpadu (viz. Obrázek 20). Jelikož převod trvá určitou dobu následuje krátká pauza (maximálně 1 sekundu). Tato pauza je realizována zavedením cyklu for. Pauzu je možné realizovat i jiným způsobem (např. pomocí Timeru). Z hlediska rozsahu a jednoduchosti je cyklus for výhodnější. Po pauze se opět provede sekvence resetu a odeslání příkazu

Match ROM. Jakmile se tato sekvence ukončí odešle se příkaz pro přečtení Scratchpadu (BEh). S tímto bytem se odešle ještě sekvence tzv. prázdných bytů (FFh). Po odeslání této sekvence převodník přečte všech 9 bytů Scratchpadu a pošle je zpět do počítače. Zde se uloží do pole bytů označené data. Přijatá data je potřeba ověřit jestli jsou přajaté bez chyby. K tomuto se používá kontrolní součet CRC. Pro realizaci CRC se vyvolá příslušná metoda obsažená v knihovně DalSemi (CRC8.compute(data, 1, 8)) a odečte se od přijatého CRC (poslední byte Scratchpadu). Je li výsledek jiný než nula vypíše se chybová hláška do bloku 1-Wire (textBox). Pokud je ale výsledek roven nule jsou data přijaté zprávně a data ze Scratchpadu je možné dále zpracovávat. Jako první se otestuje šesty byte (data[5]). Jedná se o tzv. konfigurační byte, který udává s jakým rozsahem se měřila teplota (viz. Tabulka 5).

Rozsah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Tvar (hex)	Rozlišení
9 bitů	0	0	0	1	1	1	1	1	1Fh	0.5 °C
10 bitů	0	0	1	1	1	1	1	1	3Fh	0.25 °C
11 bitů	0	1	0	1	1	1	1	1	5Fh	0.125 °C
12 bitů	0	1	1	1	1	1	1	1	7Fh	0.0625 °C

Tab.5: Konfigurační byte rozsahu teploty [3]

Po tomto testu se přepočítá teplota na základě druhého (data[1]) a třetího (data[2]) bytu. Druhý byte je tzv. LS byte a třetí byte je tzv. MS byte. MS byte navíc obsahuje informaci o znaménku.

Přepočet teploty je velice jednoduchý. Jedná se o pouhý součet MS bytu a LS bytu s následným násobením výsledku součtu a převodní konstanty. Jediným stěžením je, že se musí MS byte posunout o 8 bitů směrem k vyšším hodnotám. Přepočet teploty s kladným znaménkem probíhá dlé následujícího vzorvce:

$$tmp = ((MSB << 8) + LSB) \cdot x \qquad (1)$$

Kde

- tmp – Výsledná teplota [°C]
- LSB – LS byte (data[1])
- MSB – MS byte (data[2])
- x – Rozlišení [°C] (viz. Tabulka 5)

Při přepočtu teploty se záporným znaménkem se nejdříve provádí inverze LS a MS bytu. Po té se využívá následujícího vzorce:

$$tmp = ((invMSB << 8) + invLSB) \cdot (x \cdot (-1)) \qquad (2)$$

Kde

- tmp – Výsledná teplota [°C]
- invLSB – Inverzní LS byte (inverze bytu data[1])
- invMSB – Inverzní MS byte (inverze bytu data[2])
- x – Rozlišení [°C] (viz. Tabulka 5)

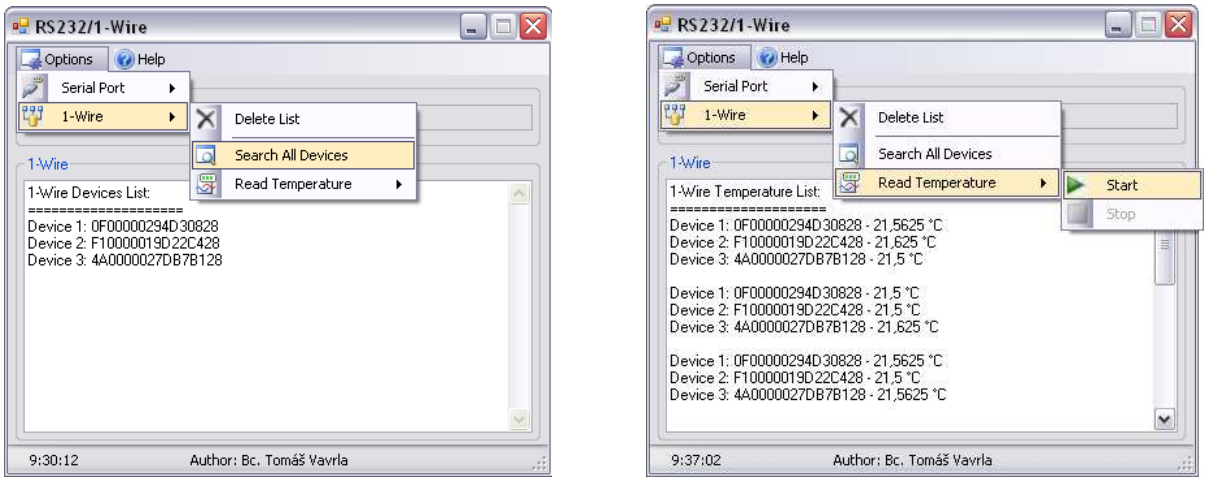
Nakonec se výsledná teplota společně s ROM kódem snímače (DS18B20) vypíše do bloku 1-Wire (textBox). Pro výpis se opět využívá metoda ToString().

Teplota	Binární tvar	Hexadecimální tvar
+125 °C	0000 0111 1101 0000	07D0h
+25.0625 °C	0000 0001 1001 0001	0550h
0 °C	0000 0000 0000 0000	0000h
-25.0625 °C	1111 1110 0110 1111	FE6Fh
-55 °C	1111 1100 1001 0000	FC90h

Tab.6: Příklad teplot pro 12 bitové rozlišení

5.2 Okno programu a jeho funkce

Oknem se ovládá celý testovací program. Okno programu se skládá z lišty menu (Options a Help), lišty s časem (Blok s časem a textem) a dvou bloků pro zobrazování informací o stavech programu (Seriál Port a 1-Wire). Celý program se tedy ovládá přes lištu menu.



Obr.21: Okno tetovacího programu a příklady činnosti

Blok označený Serial Port zobrazuje stavy sériového portu. Mezi tyto stavy patří stavy o propojení s převodníkem a stav informující, že se v daném počítači nenachází sériový port. Blok označený 1-Wire zobrazuje informace získané z 1-Wire sítě. Mezi tyto informace patří identifikační číslo zařízení (DS18B20) 1-Wire sítě a jednotlivé teploty. V dolním menu se ještě zobrazují hodiny a jméno autora. Jméno autora se změní na jinou hlášku v případě poruch nebo nějaké akce se sériovým portem. Chybová hláška se vyvolává také v podobě Message boxu což je jedna z mnoha možností podporujících vývojovým systémem Visual Studia. Message box je malé okno, které zobrazuje nějakou důležitou hlášku. Nejčastěji je využívána pro informaci o chybě. Důležitým blokem v menu je blok options. Pomocí tohoto bloku se vyvolává lišta s jednotlivými funkcemi. Základní funkce jsou dvě (Connection a Unconection) a složí pro práci se sériovou linkou. Provedou otevření (připojení) popřípadě uzavření (odpojení) sériové linky. Další funkce slouží pro práci s 1-Wire sítí. Mezi tyto funkce patří vyhledání zařízení na 1-Wire a funkce pro čtení teplot z jednotlivých čidel.

6 Realizace knihovny

Realizovaná knihovna „TempRSWire“ je vytvořena pomocí vývojového prostředí B&R Automation Studio. Vychází z testovacího programu, parametrů 1-Wire zařízení (DS18B20) a parametrů PLC automatu řady Systém X20. I přes možnost volby až z osmi programovacích jazyků (ANSI C, Automation Basic, Ladder Diagram, Strukturovaný text, atd.) je knihovna psána pomocí programovacího jazyka ANSI C. Výsledná knihovna se skládá ze tří základních bloků. První blok obsahuje funkce pro obsluhu sériového portu (RS232). Druhý blok obsahuje funkce pro komunikaci s 1-Wire sítí. Poslední třetí blok obsahuje tzv. utility. Jedná se o pomocné funkce. Mezi tyto funkce patří například funkce pro výpočet CRC kódu.

6.1 Funkce pro obsluhu sériového portu PLC automatu

Jedná se o nejzákladnější funkce, které vychází z volně dostupné knihovny DVFrame. DVFrame je knihovna, která poskytuje funkce pro obsluhu sériového portu PLC automatu (tzv. IF1 – interface 1). Jelikož tyto funkce mají podobu struktur je třeba tyto struktury kombinovat a vytvořit, tak požadovanou funkci, která bude například zapisovat nebo číst data sériové linky. Tyto struktury je třeba nějak definovat a proto je vytvořena struktura s názvem RS232. Struktura RS232 obsahuje tyto definice a také některé důležité parametry funkcí pro obsluhu 1-Wire. Kombinací struktur knihovny DVFrame jsou vytvořeny funkce uvedené v tabulce 7.

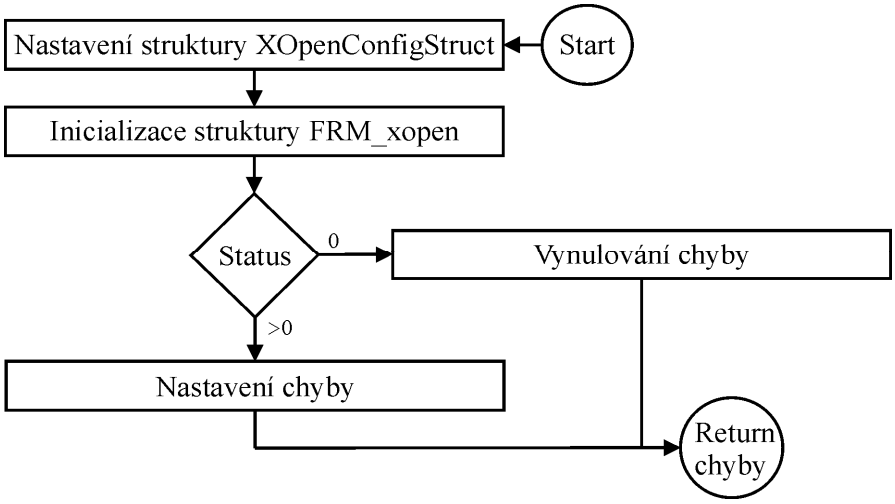
Funkce	Popis funkce
OpenCOM	Funkce pro otevření (inicializaci) sériového portu PLC automatu.
CloseCOM	Funkce pro uzavření sériového portu PLC automatu.
WriteCOM	Funkce pro zápis n bytů dat na sériovou linku.
FlushCOM	Funkce pro vyčištění všech vyrovnávacích bufferů potřebných pro sériovou komunikaci.
ReadCOM	Funkce pro čtení n bytů ze sériové linky.
SetBaudCOM	Funkce pro změnu nastavení přenosové rychlosti.

Tab.7: Funkce pro obsluhu sériového portu PLC automatu

6.1.1 Funkce OpenCOM

Tato funkce složí pro otevření (inicializaci) sériového portu programovatelného automatu. V případě PLC řady X20 se jedná o rozhraní IF1 (viz. Obrázek 3 a 17). Aby se sériový port PLC automatu otevřel, provádí se nastavení struktur XOpenConfigStruct a FRM_xopen. V případě struktury XOpenConfigStruct se jedná o pomocná konfigurační data sériového portu. V této struktuře se nastavuje počet a velikost vyrovnávacích bufferů pro příjem a vysílání. Struktura FRM_xopen provádí vlastní otevření (inicializaci) sériového portu. Pro tuto činnost vyžaduje nastavení struktury XOpenConfigStruct a konfiguraci hardwaru sériový port (FRM_xopen – mode a device). Konfigurací hardwaru sériového portu se rozumí nastavení názvu požadovaného rozhraní, parity, stop bitu, velikostí datového rámce a nastavení datové rychlosti. Konfigurace struktur XOpenConfigStruct a FRM_xopen se provádí pouhým přiřazením konkrétních hodnot. Tyto hodnoty jsou předem nastaveny na tzv. Defaultní velikosti (dáno výrobcem PLC automatu a převodníku ADA-101W). V případě

funkce OpenCOM se mění pouze nastavení názvu portu (název IF rozhraní a slotu ve kterém je rozhraní umístěno).

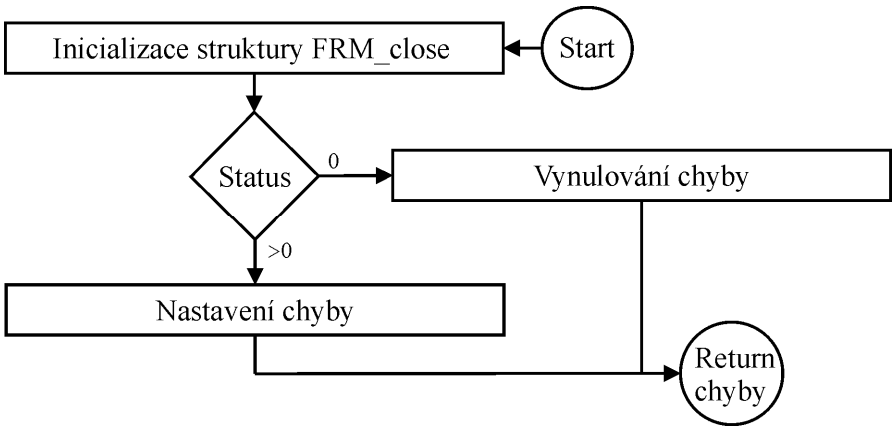


Obr.22: Diagram funkce OpenCOM

Po nastavení struktur a jejich inicializaci se ještě vyhodnocuje proměnná status. Jedná se o návratovou hodnotu struktury FRM_xopen a informuje o stavu struktury. Informuje, jestli se sériový port otevřel nebo jestli došlo k chybě. Na základě této informace dochází k návratu chybové konstanty z funkce (Return chyby).

6.1.2 Funkce CloseCOM

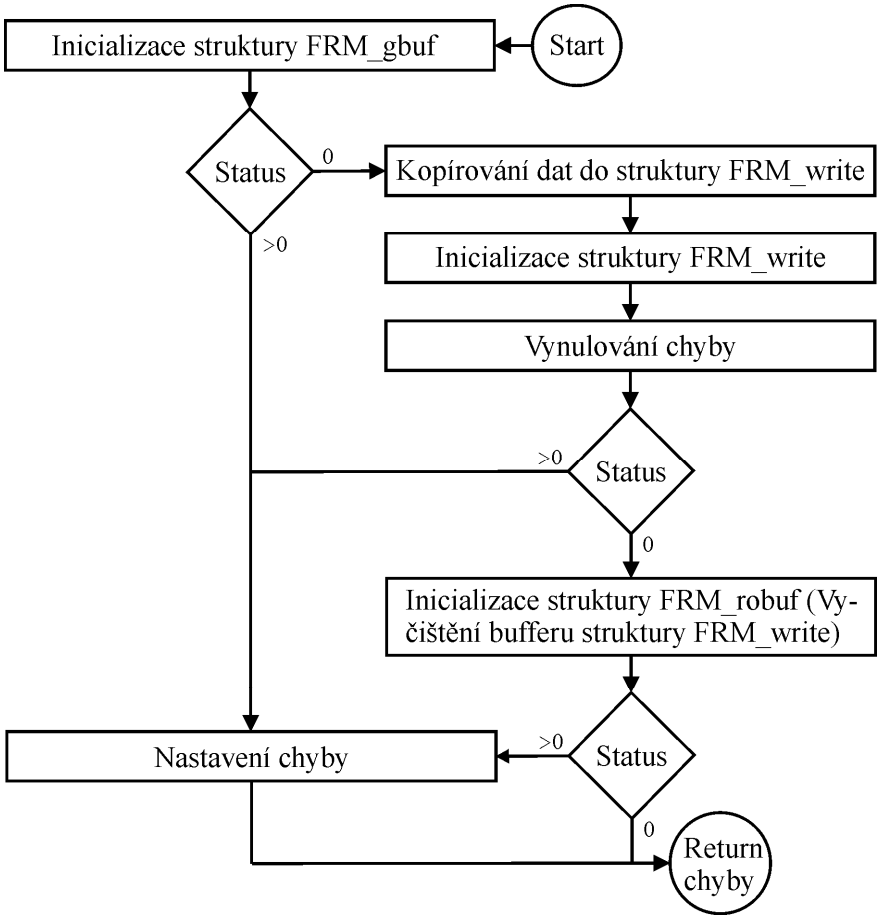
Tato funkce složí pro uzavření sériového portu programovatelného automatu. Aby se sériový port PLC automatu uzavřel, provádí se nastavení struktury FRM_close. Tato struktura vyžaduje pouhé nastavení identity (ID sériového portu PLC automatu). Hodnota identity se získává při otevření sériového portu pomocí struktury FRM_xopen. Po nastavení struktury a jejich inicializaci se ještě vyhodnocuje proměnná status. Jedná se o návratovou hodnotu struktury FRM_close a informuje o stavu struktury. Informuje, jestli se sériový port uzavřel nebo jestli došlo k chybě. Na základě této informace dochází k návratu chybové konstanty z funkce (Return chyby).



Obr.23: Diagram funkce CloseCOM

6.1.3 Funkce WriteCOM

Tato funkce složí pro zápis dat na sériovou linku. Argumenty funkce je pole s daty pro zápis na sériovou linku a proměnná udávající velikost tohoto pole. Celá funkce se skládá ze tří struktur (FRM_gbuf, FRM_robuf a FRM_write). Aby se zapsaly požadovaná data na sériovou linku, je třeba postupně vyhodnotit tyto tři struktury. Mezi každou inicializací struktury se vyhodnocuje návratová informace proměnné status. Na základě této informace se generuje chybová proměnná (return chyby).



Obr.24: Diagram funkce WriteCOM

Jako první se vyhodnocuje tzv. get buffer struktury FRM_gbuf. Tímto vyhodnocením se získává maximální velikost vyrovnávacího bufferu, odkud se zapisují data na sériovou linku. Po této inicializaci se do tohoto vyrovnávacího bufferu kopírují požadovaná data pro zápis na sériovou linku. K tomu to účelu se používá funkce memcpy.

```
memcpy(SendBuffer, Interface->WriteData, WriteBuffLen * sizeof (USINT));
```

Jako druhá se nastavuje struktura FRM_write. Do této struktury se vloží data vyrovnávacího bufferu a požadovaná šířka odesílaných dat z argumentu funkce (ta nesmí být u žádné ze struktur větší jak maximum => 256). Po nastavení struktury FRM_write se postupně zapíší data vyrovnávacího bufferu na sériovou linku a to podle přenosové rychlosti.

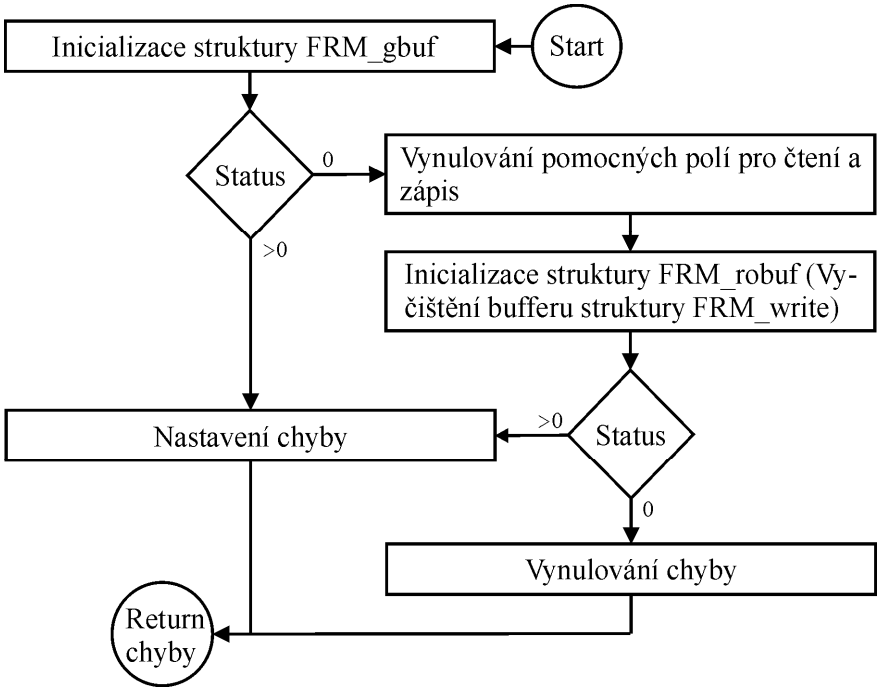
Jako třetí se nastavuje tzv. release output buffer struktury FRM_robuf. Do vyrovnávacího bufferu se nakopíruje obsah nějakého pomocného pole a tím dojde k tzv. vyčištění vyrovnávacího bufferu. Tento děj je velice důležitý, jelikož pokud by se vyrovnávací buffer nevymazal (nevyčisti) mohlo by dojít k zápisu chybných dat na sériovou linku.

6.1.4 Funkce FlushCOM

Tato funkce slouží pro vyčištění všech vyrovnávacích bufferů potřebných pro sériovou komunikaci. Celá funkce se skládá ze dvou struktur (FRM_gbuf a FRM_robuf) a tří funkcí memcpy. Funkce memcpy slouží pro kopírování prázdného pole do pomocných polí pro sériovou komunikaci. Jelikož prázdné pole neobsahuje žádná data, kopírováním se pomocná pole tzv. vyčistí (vymazání starých dat).

```
memcpy(SendBuffer, FlushData, SendBufferLength * sizeof (USINT));
memcpy(Interface->WriteData, FlushData, SendBufferLength * sizeof (USINT));
memcpy(Interface->ReadData, FlushData, 256 * sizeof (USINT));
```

Struktury FRM_gbuf a FRM_robuf plní stejnou úlohu jako ve funkci WriteCOM. To znamená, že pomocí struktury FRM_gbuf se načte vyrovnávací buffer. Následně se pomocí struktury FRM_robuf a funkce memcpy tzv. vyčistí (vymažou se stará data). Mezi každou inicializací struktury se vyhodnocuje návratová informace proměnné status. Na základě této informace se generuje chybová proměnná (return chyby).

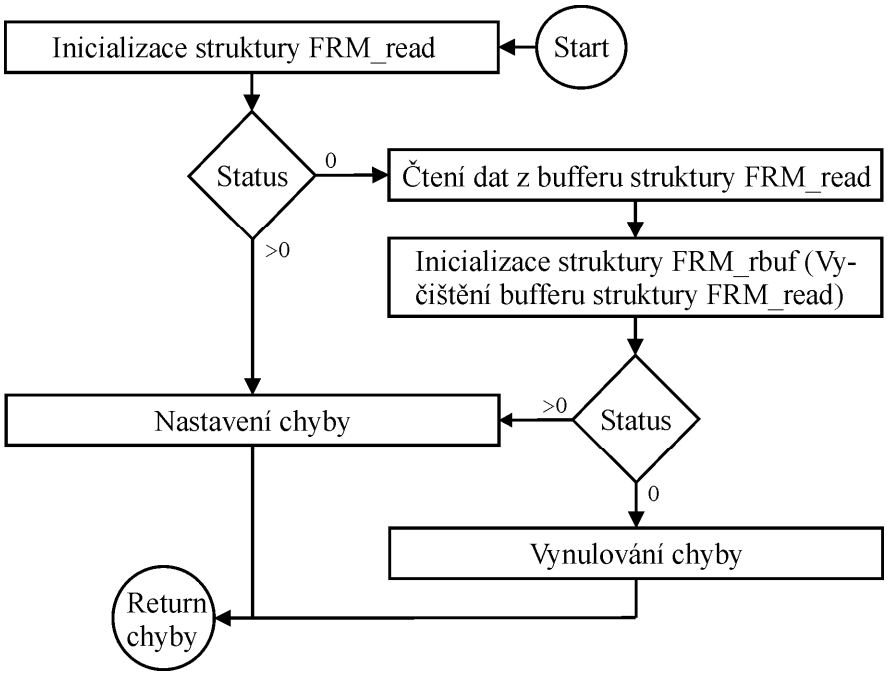


Obr.25: Diagram funkce FlushCOM

6.1.5 Funkce ReadCOM

Tato funkce složí pro čtení dat ze sériové linky. Argumenty funkce je pole s daty ze sériové linky a proměnná udávající velikost tohoto pole. Celá funkce se skládá ze dvou struktur (FRM_read a

FRM_rbuf). Aby se četly data ze sériové linky, je třeba postupně vyhodnotit tyto dvě struktury. Mezi každou inicializací struktury se vyhodnocuje návratová informace proměnné status. Na základě této informace se generuje chybová proměnná (return chyby).



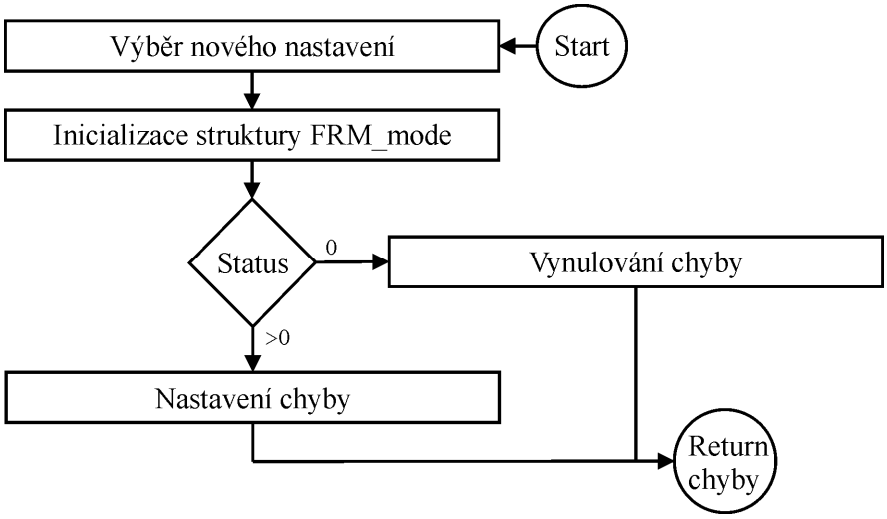
Obr.26: Diagram funkce ReadCOM

Jako první se nastavuje struktura FRM_read. Po jejím vykonání se kopírují přijaté data (pomocí funkce memcpy) z vyrovnávacího pole do pole argumentu funkce. Do pole argumentu funkce se zkopíruje pouze tolik bytů, kolik určuje argument velikosti pole.

Jako druhá se nastavuje tzv. release buffer struktury FRM_rbuf. Do struktury FRM_rbuf se vloží pomocné pole a šířka vyrovnávacího bufferu. Po inicializaci struktury FRM_rbuf dojde k tzv. vyčištění (vymazání dat) vyrovnávacího bufferu v rozsahu od prvního přijatého znaku až k poslednímu přijatému znaku.

6.1.6 Funkce SetBaudCOM

Tato funkce je určena pro změnu nastavení přenosové rychlosti. Aby se změnila přenosová rychlost sériového portu PLC automatu, provádí se nastavení struktury FRM_mode. Tato funkce slouží obecně jen ke změně nastavení. Před samotným nastavením se testuje vstupní argument funkce, na jejímž základě se do pomocného pole ukládá nové nastavení. Struktura FRM_mode kromě pole s novým nastavením vyžaduje hodnotu identity (ID sériového portu PLC automatu) získanou při otevření sériového portu. Po nastavení struktury a jejich inicializaci se ještě vyhodnocuje proměnná status. Jedná se o návratovou hodnotu struktury FRM_mode a informuje o stavu struktury. Informuje, jestli se sériový port nastavil správně nebo jestli došlo k chybě. Na základě této informace dochází k návratu chybové konstanty z funkce (Return chyby).



Obr.27: Diagram funkce SetBaudCOM

6.2 Funkce pro komunikaci s 1-Wire sítí

Jedná se o hlavní funkce, které realizují komunikaci mezi PLC automatem (Systém X20), převodníkem ADA-101W a 1-Wire sítí. Knihovna obsahuje funkce uvedené v tabulce 8. Většina těchto funkcí pracuje s již zmíněnou strukturou RS232. Součástí této struktury jsou chybové konstanty. Konstanty udávající počet nalezených zařízení, vyrovnávací buffer (pole) s ROM kódem a již zmíněné proměnné (struktury knihovny DVFrame apod.) pro práci se sériovým portem PLC automatu. Chybové proměnné informují uživatele o chybě v konkrétní funkci.

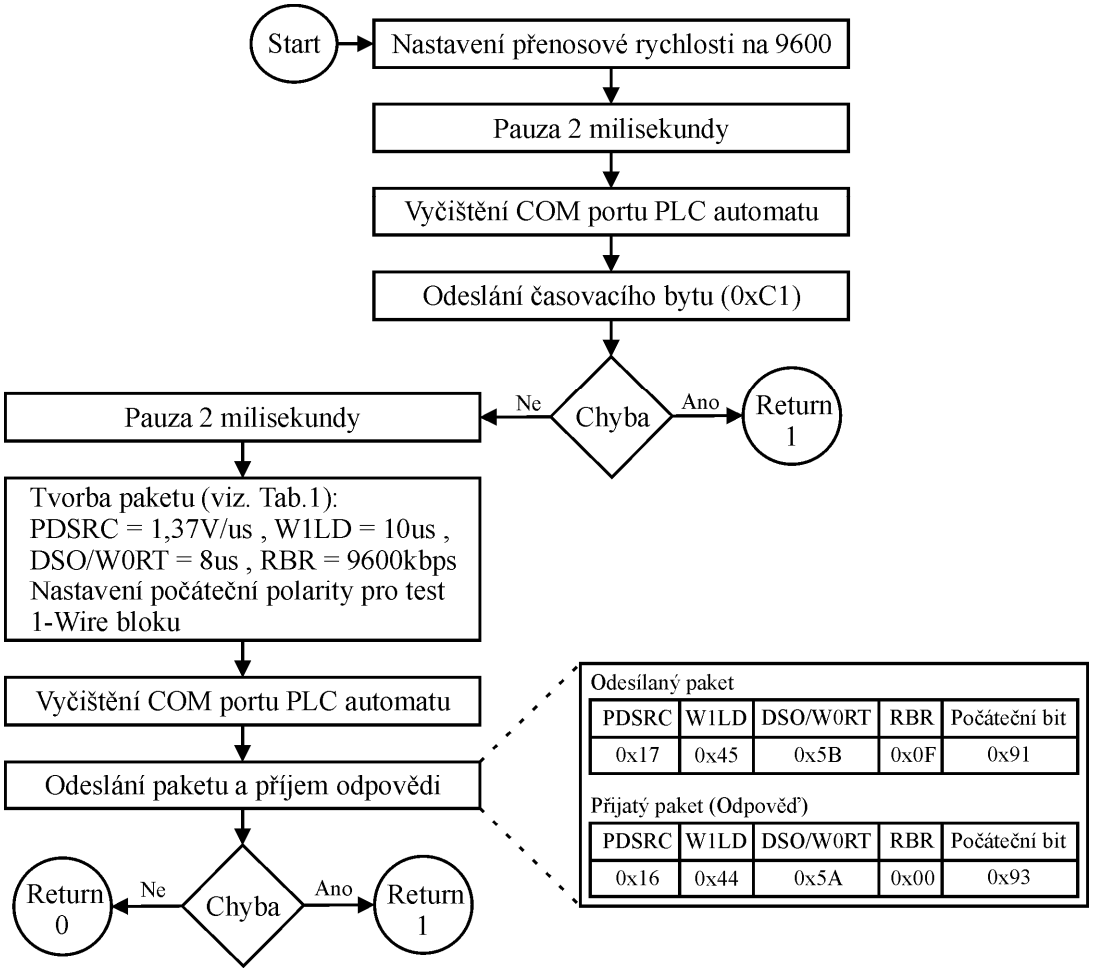
Funkce	Popis funkce
DS2480Detect	Funkce pro detekci převodníku DS2480 (ADA-101W).
OWDetect	Funkce pro detekci převodníku DS2480 a otevření sériového portu PLC automatu.
OWFindDevices	Funkce pro vyhledávání zařízení na 1-Wire síti.
OWFamilySearch	Funkce pro nastavení vyhledávacích parametrů.
OWSerialNum	Funkce pro práci s vyrovnávacím bufferem, který obsahuje sériové číslo (ROM kód).
bitacc	Funkce pro přepočet bytů na sériové číslo (ROM kód) a naopak.
OWReset	Funkce pro reset 1-Wire sítě.
OWLevel	Funkce pro nastavení napěťové úrovně na 1-Wire síti
OWNext	Funkce s vyhledávací rutinou pro převodník DS2480 (ADA-101W).
OWConvert	Funkce pro konverzi teploty a dat (přenos dat ze scratchpad paměti do EEPROM paměti => Viz. Obrázek 20).
OWMATCHROM	Funkce pro odeslání MATCH ROM příkazu a ROM kódu na 1-Wire síť.
OWSkipMATCHROM	Funkce pro odeslání Skip MATCH ROM příkazu na 1-Wire síť.
OWWriteConfig	Funkce pro odeslání konfiguračních dat na 1-Wire zařízení dané ROM kódem.
OWWriteConfigAll	Funkce pro odeslání konfiguračních dat na všechna zařízení umístěných na 1-Wire síti.
OWReadScratchpad	Funkce pro čtení scratchpad paměti z 1-Wire zařízení dané ROM kódem.
OWCopyScratchpad	Funkce pro kopírování dat ze scratchpad paměti do EEPROM paměti. U 1-Wire zařízení daného ROM kódem.
OWCopyScratchpadAll	Funkce pro kopírování dat ze scratchpad paměti do EEPROM paměti.

	Pro všechny zařízení na 1-Wire síti.
OWPresentDevice	Funkce provádějící test přítomnosti zařízení na 1-Wire síti.
OWReadTemp	Funkce pro čtení teploty.
OWCalculationTemp	Funkce pro přepočet teploty.
OWMeasureTemp	Funkce řešící celý proces měření a vyhledávání zařízení na 1-Wire síti.

Tab.8: Funkce pro komunikaci s 1-Wire sítí

6.2.1 Funkce DS2480Detect a OWDetect

Tyto funkce detekují převodník DS2480 (ADA-101W). Jedná se tedy o nejdůležitější funkce z hlediska komunikace mezi PLC automatem a 1-Wire sítí. Kromě detekce převodníku provádějí i prvotní konfiguraci zmíněného převodníku. Převodník se musí konfigurovat, jelikož neobsahuje oscilátor. Pro tento účel se jako první nastavuje přenosová rychlost (viz. Obrázek 28) na základní úroveň a minimálně po 2 milisekundách se odesílá tzv. časovací byte. [11]

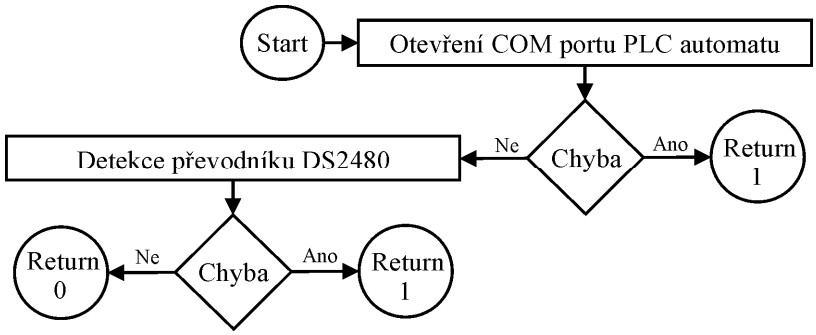


Obr.28: Diagram funkce DS2480Detect [11]

Po časovacím bytu následuje další 2 milisekundová pauza pro zotavení převodníku. Pro dokončení konfigurace se ještě odesílá paket s 5 byty. Byty PDSRC, W1LD, DSO/W0RT a RBR (viz. Tabulka 1) konfigurují 1-Wire síť a jsou nastaveny na doporučené hodnoty. Tyto doporučené hodnoty nastavují tzv. flexible speed režim [11]. Jedná se o nastavení pro komunikaci na delší vzdálenosti (dle výrobce

na vzdálenost větší jak 100 metrů). Nejdůležitější byte je poslední. Tento byte nastavuje počáteční polaritu 1-Wire sítě a zároveň testuje převodník. Je li přítomen převodník odesílá na základě zmíněných 5 bytů odpověď. Odpověď musí mít tvar uvedený na obrázku 28. V tomto tvaru je nejdůležitější právě 5 byte informující o stavu převodníku. Na základě tohoto 5 bytu a 4 bytu (byte přenosové rychlosti) se generuje chybová konstanta (Return 0 nebo 1).

Funkce OWDetect pouze rozšiřuje účel funkce DS2480Detect o otevření a inicializaci sériového portu PLC automatu. S tím souvisí i vyhodnocení chyby (viz. Obrázek 29).

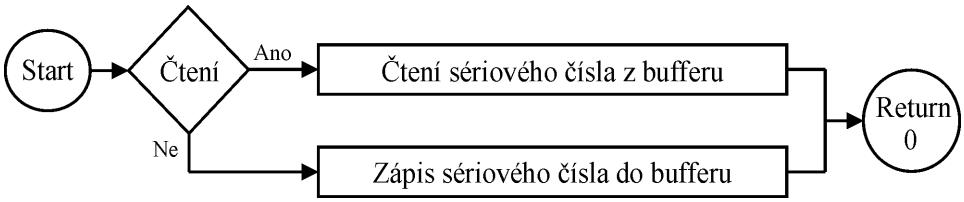


Obr.29: Diagram funkce OWDetect

6.2.2 Funkce OWFamilySearch, OWSerialNum a bitacc

Funkce OWFamilySearch, OWSerialNum a bitacc jsou pomocné funkce. Funkce OWFamilySearch slouží pro počáteční nastavení vyhledávacích parametrů. Mezi tyto parametry patří family kód (viz. Obrázek 8) hledaného zařízení, který se přiřadí do vyrovnávacího bufferu (pole) se sériovým číslem (ROM kód). Tato funkce je součástí funkce pro hledání zařízení na 1-Wire síti (OWFindDevices).

OWSerialNum je funkce realizující přístup do vyrovnávacího bufferu (pole) s ROM kódem. Pro tento účel se nejprve testuje vstupní argument funkce pro čtení (viz. Obrázek 30). Po tomto testu se buď do vyrovnávacího buferu zapisuje nebo se z něj čte. Zápis i čtení se realizuje funkcí for. Pomocí této funkce se prochází jednotlivé prvky bufferu.

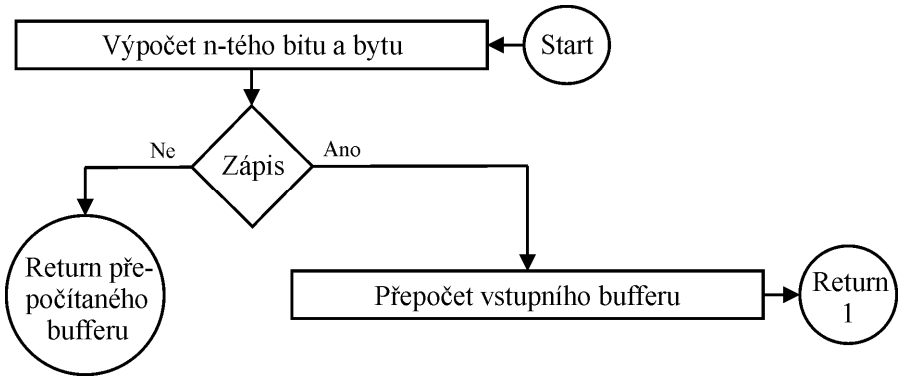


Obr.30: Diagram funkce OWSerialNum

Při realizaci funkce bitacc se vychází z vlastností převodníku DS2480 uvedených v aplikačním listě výrobce [10, 11]. Funkce slouží pro přepočet bytů na ROM kód a naopak. Tato funkce je součástí funkce pro hledání zařízení na 1-Wire síti (OWFindDevices) a je její nejdůležitější částí.

V první části této funkce se vypočítává n-tý byte a n-tý bit třetího argumentu funkce. N-tý byte má funkci ukazatele na vstupní buffer. N-tý bit má funkci konstanty pro přepočet vstupního bufferu. Po

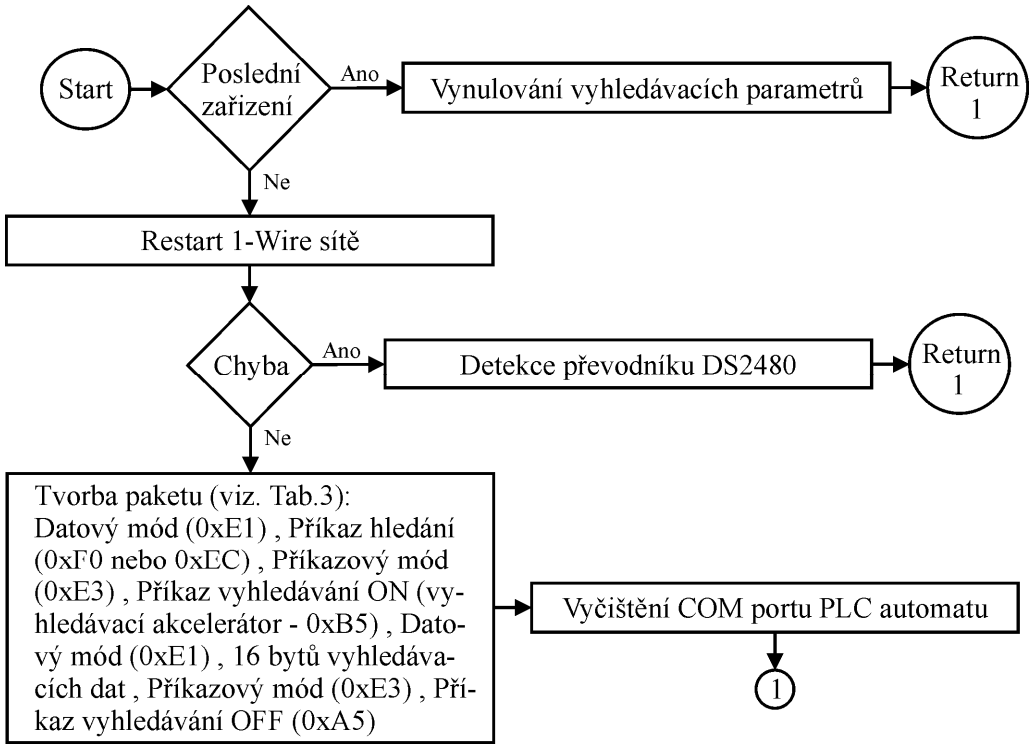
tomto výpočtu se stejně jako u funkce OWSerialNum testuje vstupní argument funkce pro čtení nebo zápis (viz. Obrázek 31). Na základě tohoto testu se následně přepočítává n-tý byte vstupního bufferu (pole) argumentu funkce. V případě zápisu se výsledek přepočtu ukládá zpět do vstupního bufferu argumentu funkce. V případě čtení se výsledek převodu vrací pomocí příkazu return.



Obr.31: Diagram funkce bitacc

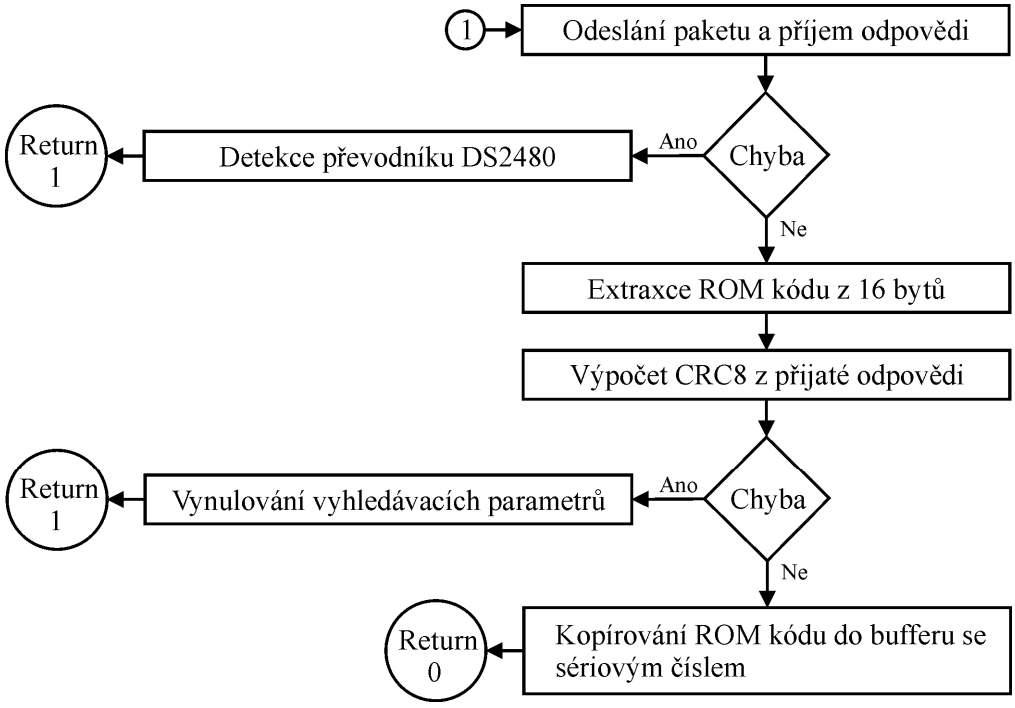
6.2.3 Funkce OWFindDevices a OWNext

Tyto dvě funkce slouží pro vyhledávání zařízení na 1-Wire síti. Funkce OWNext tvoří hlavní část vyhledávacího algoritmu. V první části této funkce se testuje příznak posledního zařízení, které lze na 1-Wire síti nalézt. Pokud je nalezeno poslední zařízení vynulují se příznaky vyhledávání a ukončí se funkce (Return 1).



Obr.32: Diagram první části funkce OWNext

Po testu posledního zařízení se volá funkce resetu 1-Wire síti. Pokud se funkce vykoná správně pak je1-Wire síť připravena pro vyhledávání. V případě chyby se volá funkce DS2480Detect, která plní úlohu synchronizace. Po resetu se vytváří paket dat. První část paketu obsahuje byty příkazů (viz. Obrázek 32). Druhá část paketu je tvořena 16 byty vyhledávacích dat, které se získávají pomocí funkce bitacc (viz. Funkce bitacc). Jako vstupní informace pro funkci bitacc slouží nulový ROM kód vytvořený pomocí funkce OWFamilySearch.

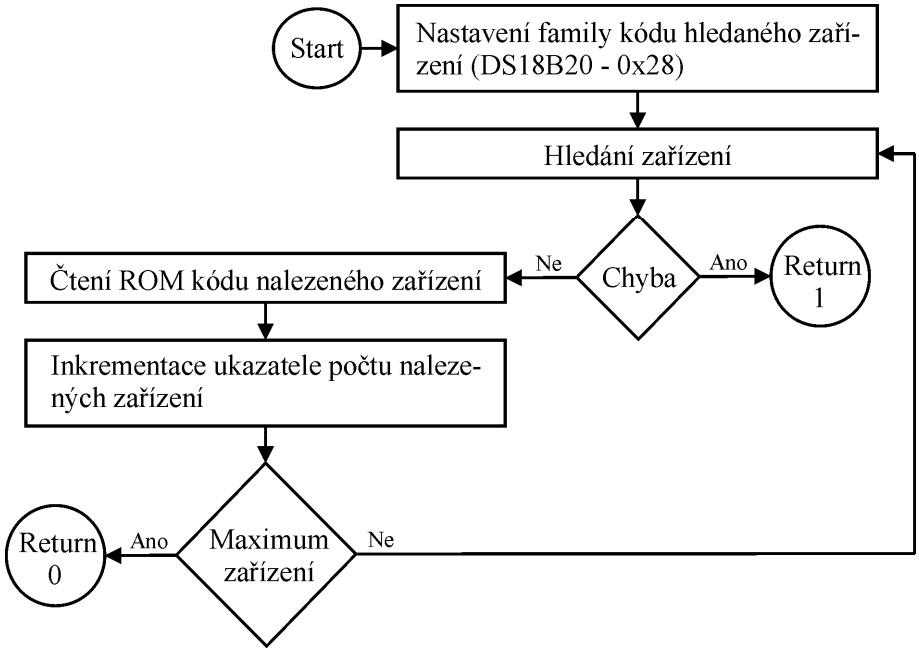


Obr.33: Diagram druhé části funkce OWNext

Druhá část obsahuje rutiny pro odeslání paketu, příjem odpovědí a její zpracování. Odpověď má podobu 17 bytů. První byte obsahuje příkaz vyhledávání. Tímto bytem se nastavuje režim vyhledávání (DS2480 rozlišuje dva režimy hledání => Hledání zařízení F0h a hledání alarmů ECh => Volí se přes argument funkce). Za prvním bytem následuje 16 bytů informace o ROM kódu. Tato informace se opět přepočítává (extrahuje) pomocí funkce bitacc. Extrahovaná data mají podobu 8 bytů a odpovídají ROM kódu nalezeného zařízení. 8 bytová informace se dále kontroluje pomocí CRC. Postupně se vypočítává 8 bitová CRC hodnota každého bytu získaného ROM kódu. Pokud je výsledná CRC hodnota rovna nule uloží se ROM kód do bufferu sériového čísla (pomocí funkce OWSerialNum) a funkce se ukončí (Return 0). Pokud, ale nastane chyba (chybné CRC) vynulují se příznaky vyhledávání a funkce se ukončí (Return 1). V případě chyby příjmu se vyvolá funkce DS2480Detect pro synchronizaci.

Funkce OWFindDevices je hlavní funkce pro vyhledávání zařízení na 1-Wire síti (viz. Obrázek 34). Tato funkce slučuje všechny funkce vyhledávání. Jako první se volá již zmíněná funkce (OWFamilySearch) pro nastavení family kódu do algoritmu vyhledávání. Po tomto nastavení se volá funkce OWNext, která provede vyhledávací proces. Následně se testuje návratová hodnota. V případě detekce chyby se funkce ukončí (Return 1). V opačném případě se volá funkce (OWSerialNum) pro

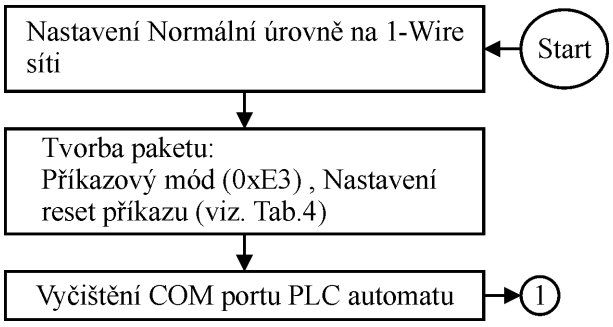
získání ROM kódu z bufferu sériového čísla. ROM kód se otestuje na family kód zadaný přes argument funkce. V případě shody se přičítá jednička do argumentu funkce ukazatele počtu nalezených zařízení a současně se přiřadí ROM kód do pole ROM kódu argumentu funkce. Než se funkce ukončí, ještě se testuje maximální počet zařízení nalezených na 1-Wire. K tomuto účelu slouží klasická funkce if a argument funkce s maximálním počtem zařízení. Pokud je nalezeno málo zařízení volá se opět funkce OWNext. Při opaku se funkce ukončí (Return 0).



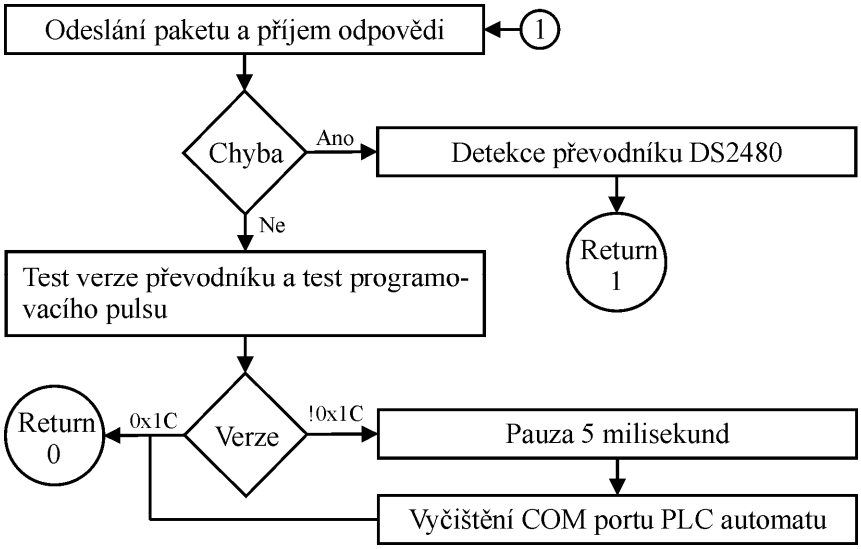
Obr.34: Diagram funkce OWFindDevices

6.2.4 Funkce OWReset

Tato funkce složí pro reset 1-Wire sítě (viz. Obrázek 35 a 36). Ve funkci se jako první testuje úroveň na 1-Wire síti. K tomu to účelu slouží funkce OWLevel. Po tomto testu se vytváří paket dat. Do proměnné paket se vloží byte příkazového režimu a byte příkazu Reset. Pokud nenastane chyba, paket se zapíše na sériovou linku. V případě chyby se volá funkce DS2480Detect pro synchronizaci. Odpověď má podobu 1 až 2 bytů. Na jejich základě se vyčítá informace o verzi převodníku a informace o programovacím pulzu. Pokud není verze rovná hodnotě 1Ch volá se funkce pauzy (dle výrobce je minimálně 5 milisekund) a funkce se ukončí (Return 0).



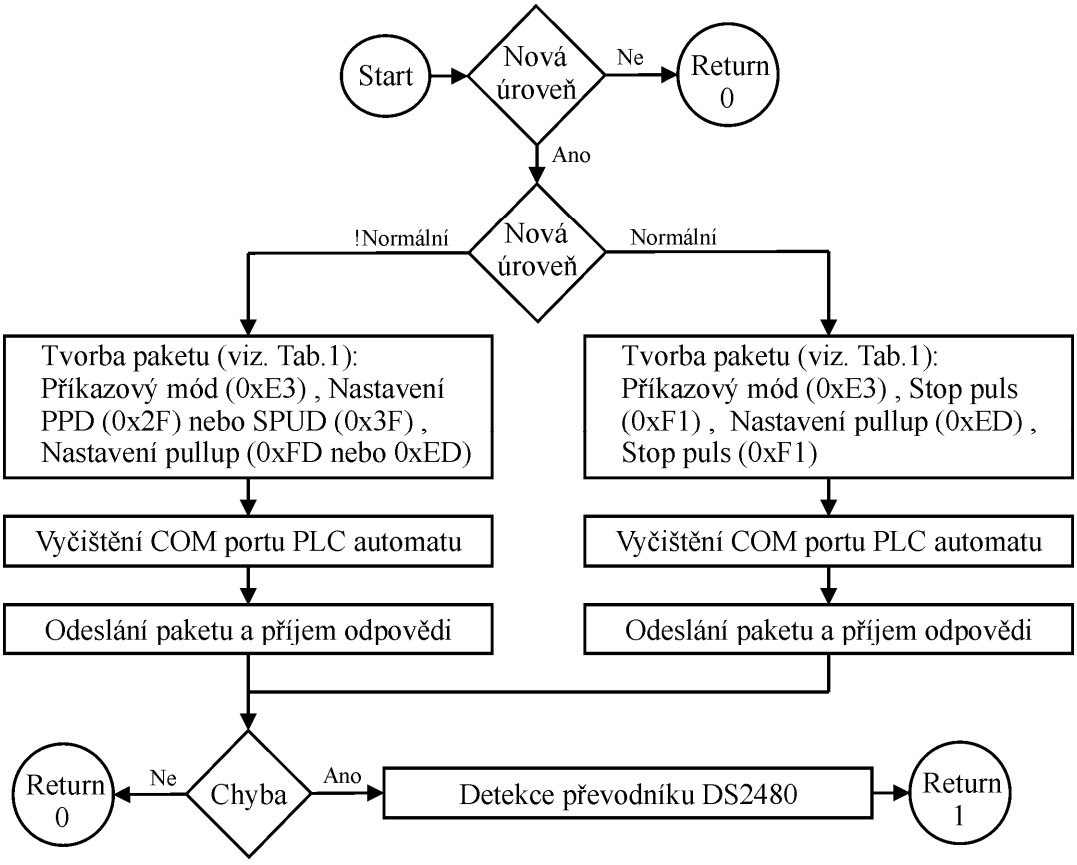
Obr.35: Diagram první části funkce OWReset



Obr.36: Diagram druhé části funkce OWReset

6.2.5 Funkce OWLevel

Jedná se o pomocnou funkci pro nastavení napěťové úrovně na 1-Wire síti (viz. Obrázek 37). Napěťová úroveň se zadává do proměnné argumentu funkce. Pokud je nová úroveň stejná jako stávající, funkce se ukončí (Return 0).

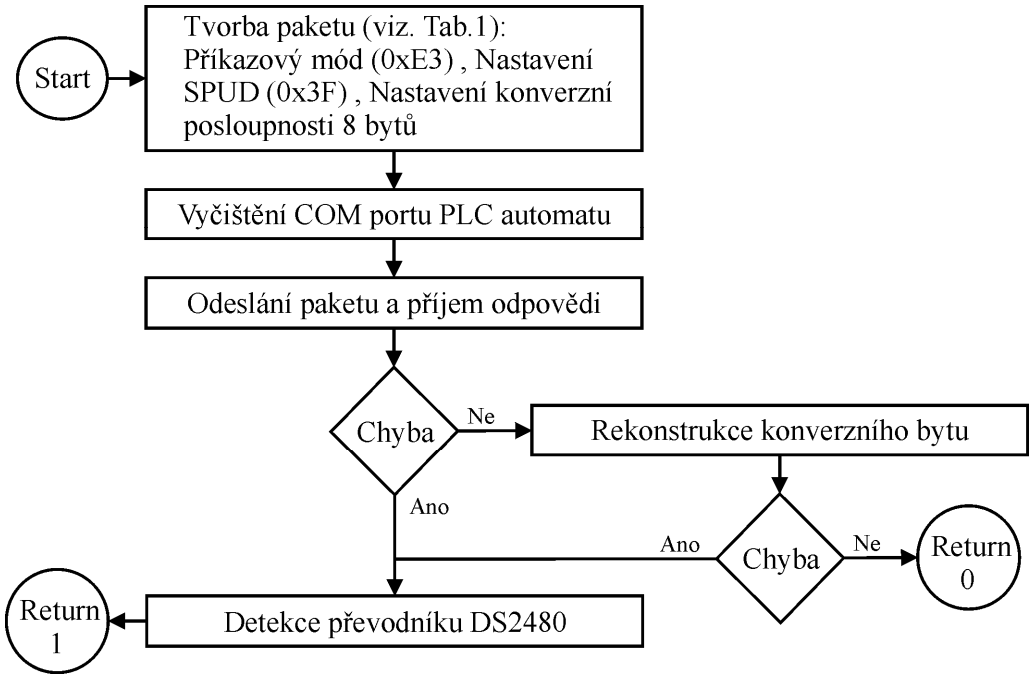


Obr.37: Diagram funkce OWLevel

Na základě tzv. normální úrovně se vytváří paket dat. Paket obsahuje 4 byty. První byte je vždy příkazový. V případě normální úrovně jsou další 3 byty pro nastavení Pullup na 5 voltovou úroveň. V případě, že se nejedná o normální úroveň, jedná se o tzv. programovací úroveň. 3 byty slouží pro nastavení Pullup na 12 voltovou úroveň. Tato úroveň lze použít jen v případě, že je aktivován přepínač na přední straně převodníku ADA-101W jinak je na 1-Wire síti tzv. normální úroveň (5 voltů). Jako odpověď se přijímají 2 byty. Tyto 2 byty by měly mít určitou formu. Tato forma se testuje v podobě příkazu if. Pokud je výsledek testu správný funkce se ukončí (Return 0). Pokud je výsledek chybný vykoná se ještě funkce DS2480Detect pro synchronizaci (vrací se příznak chyby => Return 1).

6.2.6 Funkce OWConvert

Jedná se o funkci, která generuje konverzní posloupnost pro převod teploty nebo pro kopírování dat ze scratchpad paměti do EEPROM paměti. Funkce (viz. Obrázek 38) začíná tvorbou paketu. Paket obsahuje příkazový byte, byte SPUD (viz. Tabulka 1) a 8 bytů konverzní posloupnosti. 8 bytová posloupnost se vypočítává z konverzního příkazu zadaného přes argument funkce. Po vytvoření paketu se paket odesílá na sériovou linku.

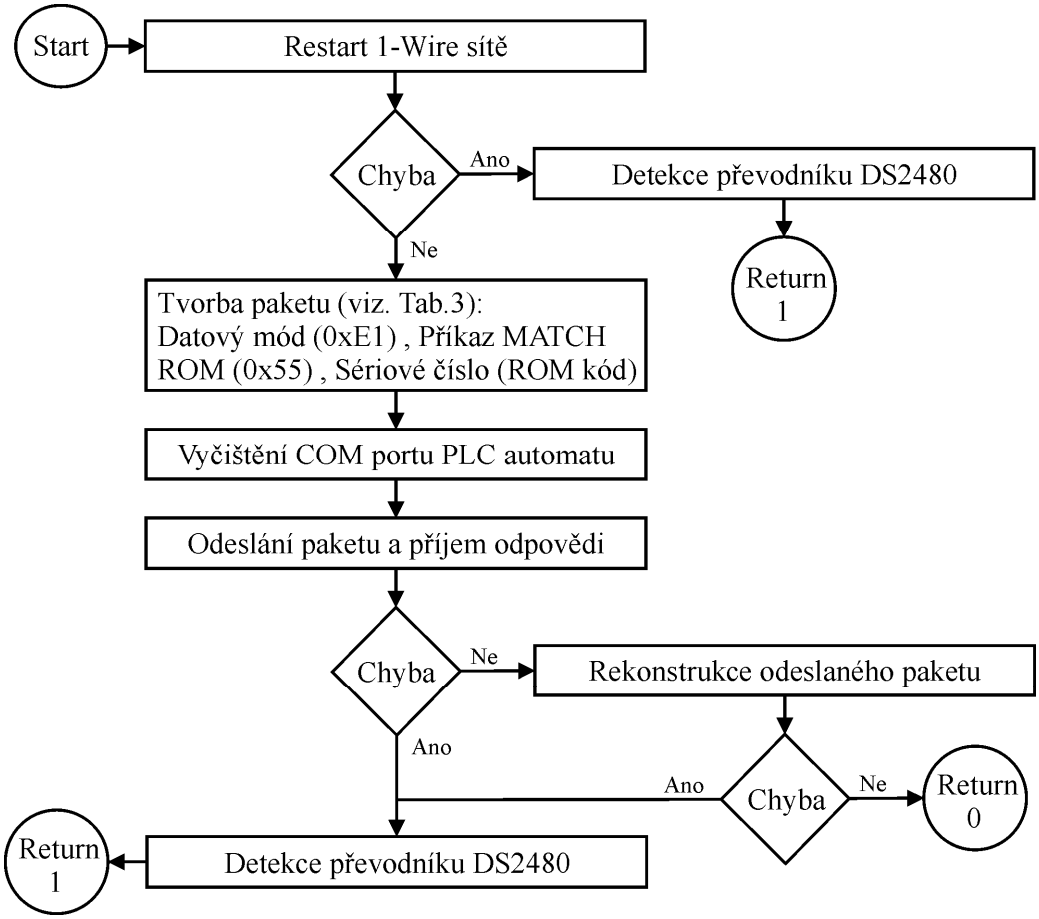


Obr.38: Diagram funkce OWConvert

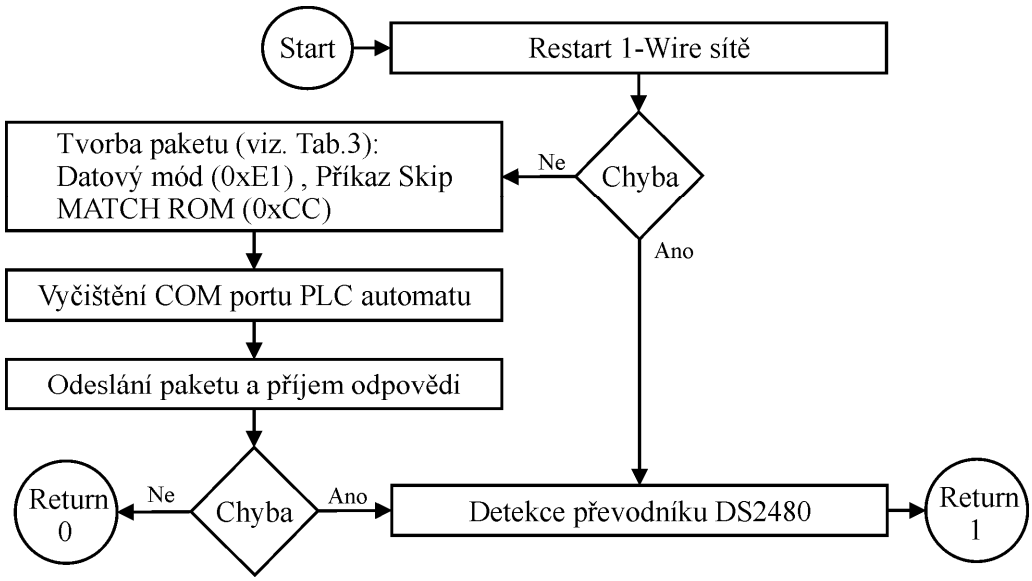
Převodník jako odpověď posílá stejnou posloupnost, jakou obsahuje vytvořený paket, ale bez příkazového bytu. Pro účel testu pak postačuje zpětně převést konverzní posloupnost na jeden byte. Pokud je rekonstruovaný byte roven konverznímu bytu z argumentu funkce, chyba nenastala a funkce se ukončí (Return 0). V případě rozdílného výsledku se provede ještě funkce DS2480Detect pro synchronizaci (vrací se příznak chyby => Return 1).

6.2.7 Funkce OWMATCHROM a OWSkipMATCHROM

Tyto dvě funkce slouží pro zpřístupnění jednoho nebo všech zařízení nalezených na 1-Wire síti. Obě funkce jsou přibližně stejné. Jako první se v obou funkcích volá funkce resetu (OWReset).



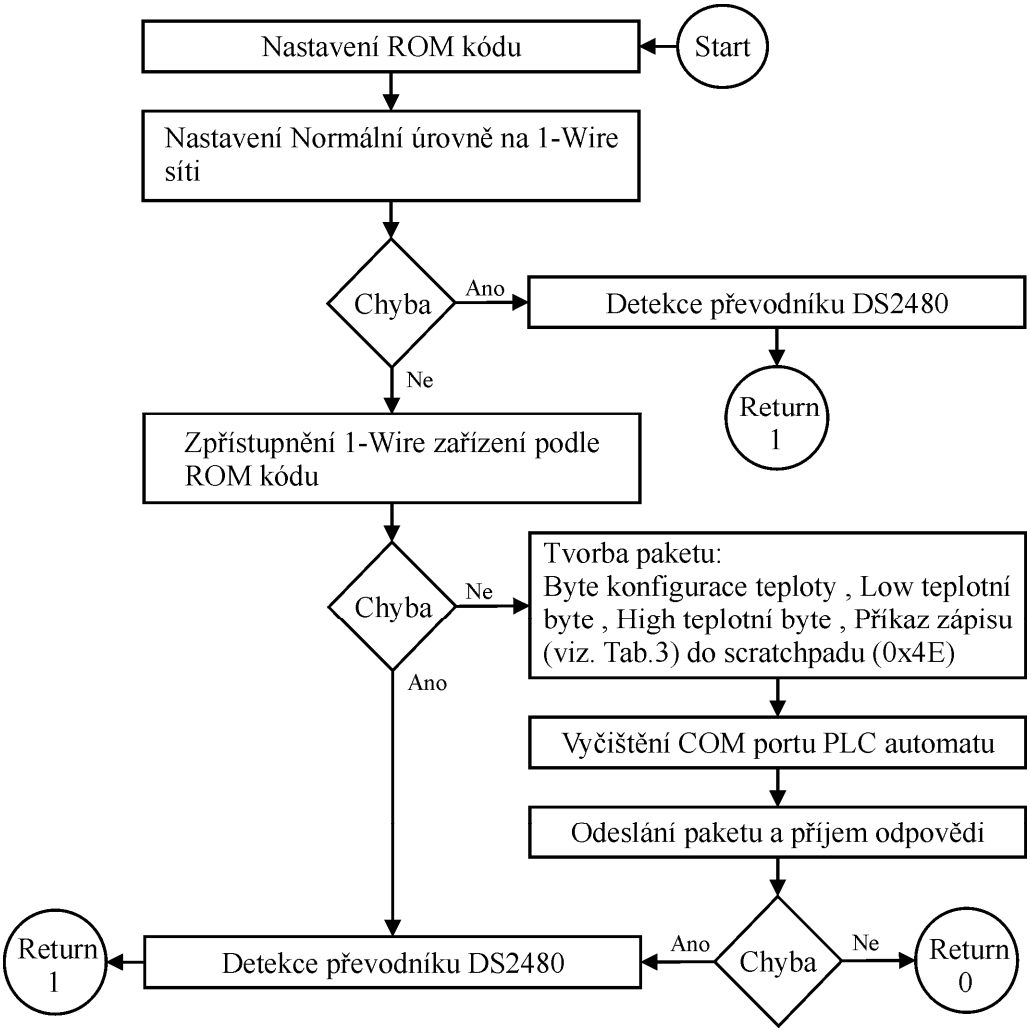
Obr.39: Diagram funkce OWMATCHROM



Obr.40: Diagram funkce OWSkipMATCHROM

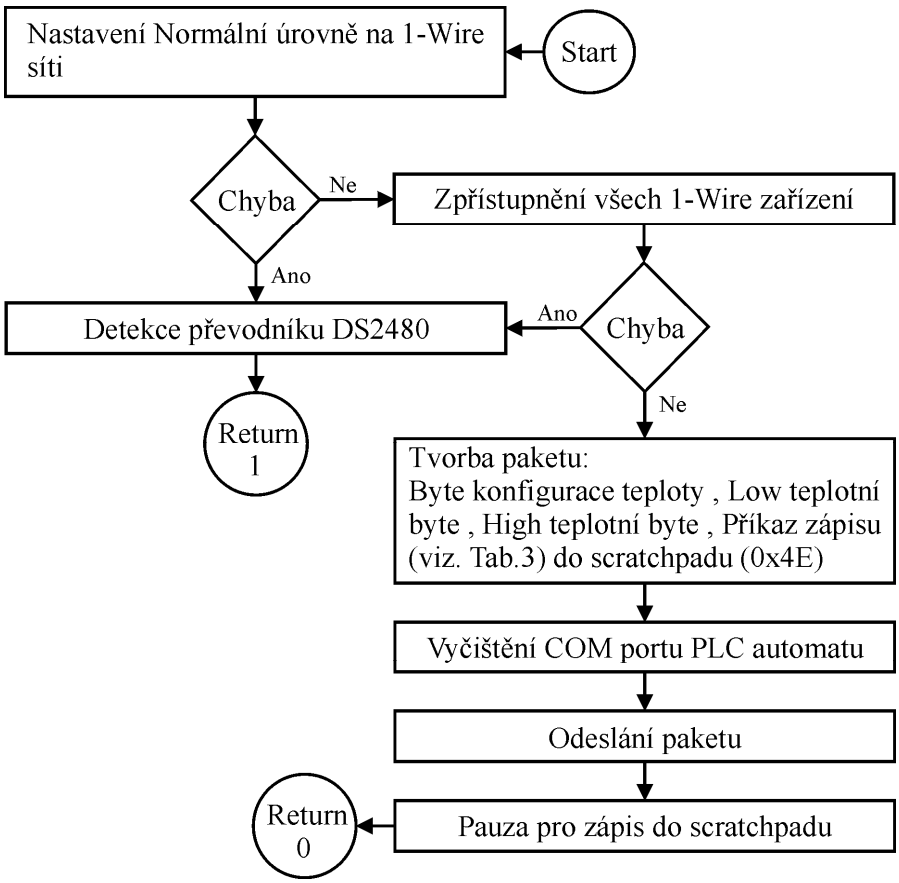
Po resetu následuje hlavní část obou funkcí. A to tvorba paketu dat. V případě funkce OWMATCHROM paket obsahuje 10 bytů (viz. Obrázek 39) a v případě OWSkipMATCHROM funkce paket obsahuje pouze 2 byty (viz. Obrázek 40). První byte je pro obě funkce stejný. Jedná se o byte pro nastavení datového režimu. Druhý byte je příkazový a nastavuje režim Match ROM a Skip Match ROM (viz. Tabulka 3). Dalších 8 bytů obsahuje pouze funkce OWMATCHROM. Jedná se o byty obsahující sériové číslo (ROM kód). ROM kód se do paketu vkládá přes příkaz for. Po vytvoření paketu se paket odesílá na sériovou linku (funkce WriteCOM). Převodník jako odpověď posílá stejnou posloupnost, jakou obsahuje vytvořený paket, ale bez prvního bytu pro datový režim. To znamená, že chyba u obou funkcí nastává, pokud se vyhodnocená odpověď neshoduje s odesílaným paketem. V případě OWMATCHROM funkce se navíc zpětně získává ROM kód. Opět pomocí příkazu for. Při detekci chyby se u obou funkcí volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1). Nenastane li chyba, funkce se ukončují a vrací nulový příznak chyby (Return 0).

6.2.8 Funkce OWWriteConfig a OWWriteConfigAll



Obr.41: Diagram funkce OWWriteConfig

Tyto dvě funkce slouží pro odesílání konfiguračních dat na jedno nebo všechny zařízení 1-Wire sítě. Obě funkce jsou přibližně stejné a jsou velice jednoduché. V první části se u obou funkcí volají funkce pro nastavení úrovně (OWLevel) 1-Wire sítě a funkce pro zpřístupnění jednoho nebo všech zařízení. V případě funkce OWWriteConfig (viz. Obrázek 41) se jedná o funkci OWMATCHROM a funkci pro vložení ROM kódu (OWSerialNum) do bufferu sériových čísel. Konfigurační data se zapisují pouze na zařízení s odpovídajícím ROM kódem z pole argumentu funkce. V případě funkce OWWriteConfigAll (viz. Obrázek 42) se jedná pouze o funkci OWSkipMATCHROM. Konfigurační data se zapisují do všech zařízení přítomných na 1-Wire síti.

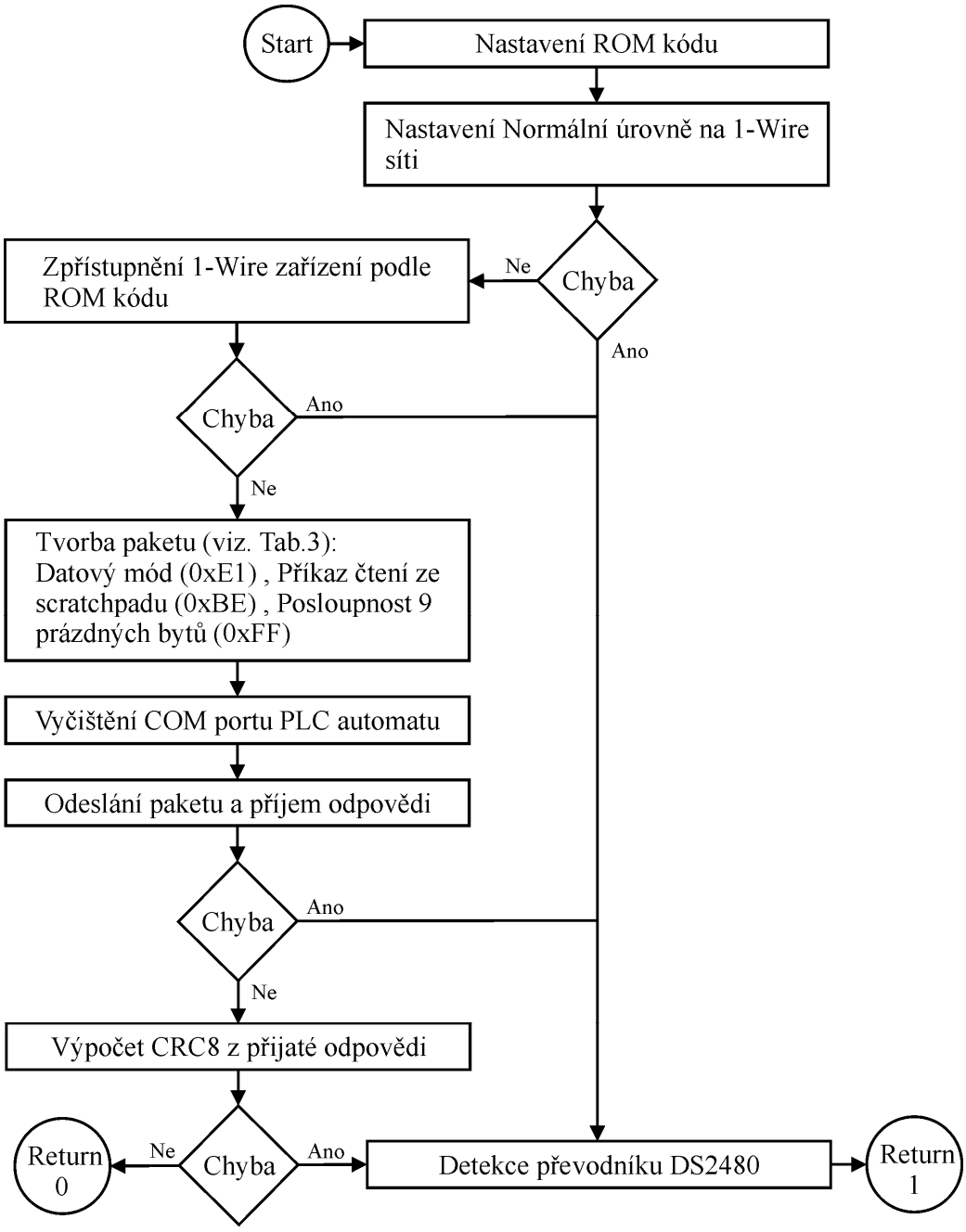


Obr.42: Diagram funkce OWWriteConfigAll

V druhé části se vytváří paket dat. Paket je stejný pro obě funkce. Obsahuje 4 byty. První byte obsahuje příkaz pro zápis (viz. Tabulka 3) bytů paketu do scratchpad paměti. Další tři byty jsou byty proměnných argumentu funkce. Jedná se o alarmové a konfigurační hodnoty rozsahu teploty (viz. Tabulka 5). Po vytvoření paketu se paket odesílá na sériovou linku (funkce WriteCOM). Převodník v případě funkce OWWriteConfig posílá odpověď v podobě bytu, který je roven prvnímu bytu odesílaného paketu. Funkce OWWriteConfigAll nepřijímá žádnou odpověď, jelikož se paket odesílá na všechna zařízení 1-Wire sítě. S tím souvisí nutnost zavedení zpoždění. Realizuje se inkrementací pomocné proměnné a srovnáním výsledku s argumentem funkce zpoždění. Při detekci chyby v nějaké části funkcí se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1). Nenastane li chyba, funkce se ukončují a vrací nulový příznak chyby (Return 0).

6.2.9 Funkce OWReadScratchpad

Jedná se o funkci pro čtení scratchpad paměti 1-Wire zařízení. V první části (viz. Obrázek 43) se volá funkce pro nastavení úrovně (OWLevel) 1-Wire sítě a funkce pro zpřístupnění zařízení (OWMATCHROM). Pro zpřístupnění zařízení obsahuje funkce argument pole s ROM kódem. ROM kód argumentu funkce se pomocí funkce OWserialNum vkládá do bufferu sériových čísel.



Obr.43: Diagram funkce OWReadScratchpad

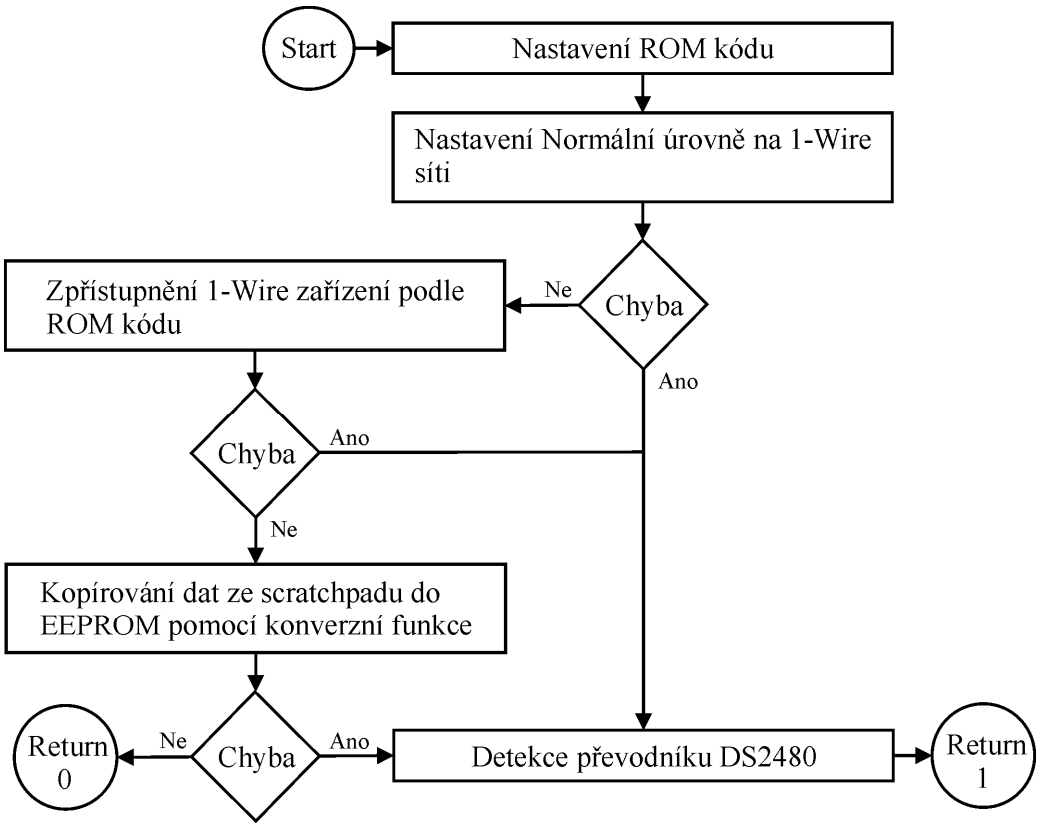
V druhé části se vytváří paket dat. Paket obsahuje 11 bytů. První byte nastavuje datový režim. Druhý byte obsahuje příkaz (viz. Tabulka 3) pro čtení dat ze scratchpad paměti. Posledních 9 bytů jsou

tzv. prázdné byty. Prázdné byty se do paketu umisťují z důvodu přijímané odpovědí. Převodník zaregistruje, že má kromě odpovědi na příkaz odeslat i konkrétní posloupnost dat scratchpad paměti.

Převodník jako odpověď posílá stejnou posloupnost, jakou obsahuje vytvořený paket, ale bez prvního bytu. To znamená, že chyba nastává, pokud se první byte neshoduje s druhým bytem vytvořeného paketu. Horních 9 bytů nese informaci o scratchpad paměti. Za tímto účelem se provádí kontrola pomocí CRC výpočtu. Postupně se vypočítává 8 bitová CRC hodnota každého bytu informace o scratchpad paměti. Pokud je výsledná CRC hodnota rovna nule vloží se výsledek (pomocí funkce memcpy) do pole argumentu funkce. Při detekci chyby v nějaké části funkce (i při výpočtu CRC) se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1). Nenastane li chyba, funkce se ukončí a vrací nulový příznak chyby (Return 0).

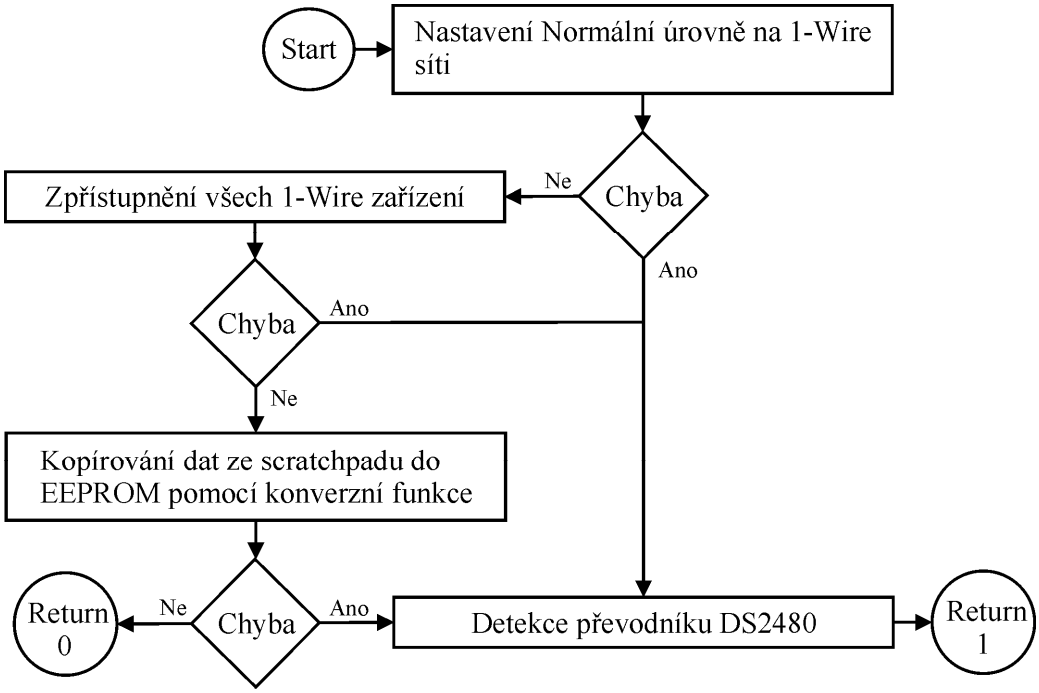
6.2.10 Funkce OWCopyScratchpad a OWCopyScratchpadAll

Tyto dvě funkce slouží pro kopírování konfiguračních dat ze scratchpad paměti do EEPROM paměti. Obě funkce jsou přibližně stejné. Jako první se v obou funkcích volají funkce pro nastavení úrovně (OWLevel) 1-Wire sítě a funkce pro zpřístupnění jednoho nebo všech zařízení. V případě funkce OWCopyScratchpad (viz. Obrázek 44) se jedná o funkci OWMATCHROM a funkci pro vložení ROM kódu (OWSerialNum) do bufferu sériových čísel. Scratchpad data se kopírují pouze na 1-Wire zařízení s odpovídajícím ROM kódem z pole argumentu funkce. V případě funkce OWCopyScratchpadAll (viz. Obrázek 45) se jedná pouze o funkci OWSkipMATCHROM. Scratchpad data se kopírují do všech zařízení přítomných na 1-Wire síti.



Obr.44: Diagram funkce OWCopyScratchpad

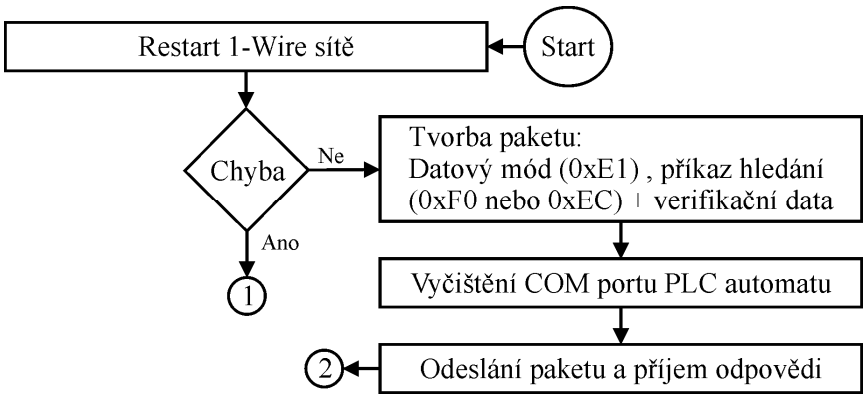
Po zpřístupnění 1-Wire zařízení se volá konverzní funkce (OWConvcert). Do argumentu této funkce se vkládá příkazový byte (48h) pro kopírování scratchpad dat do EEPROM paměti (viz. Tabulka 3). Při detekci chyby v nějaké části funkcí se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1). Nenastane li chyba, funkce se ukončují a vrací nulový příznak chyby (Return 0).



Obr.45: Diagram funkce OWCopyScratchpadAll

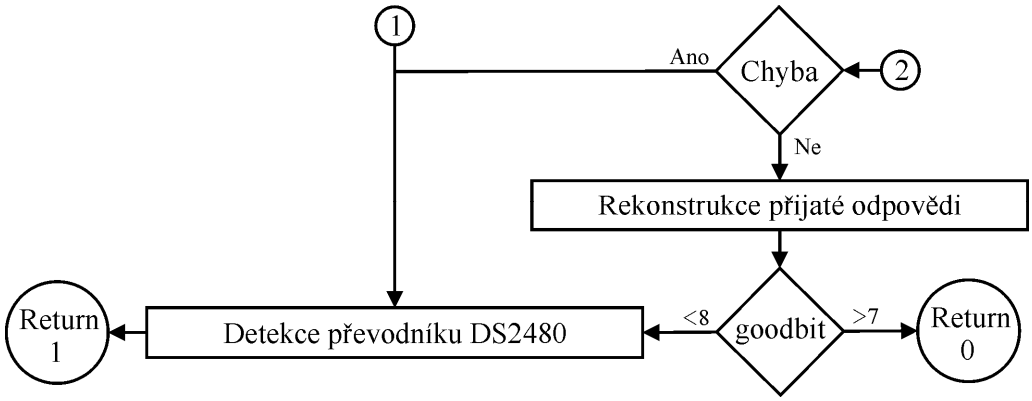
6.2.11 Funkce OWPresentDevice

Tato funkce provádí test přítomnosti zařízení na 1-Wire síti. V první části funkce (viz. Obrázek 46) se volá funkce resetu (OWReset). Po resetu následuje hlavní část funkce. A to tvorba paketu dat. Po resetu se vytváří paket dat. Paket obsahuje dva příkazové byty a 23 vyhledávacích bytů tzv. verifikační data. Prvním bytem je příkaz pro nastavení datového režimu. Druhým bytem (Hledání zařízení F0h a hledání alarmů ECh) je vyhledávací příkaz (viz. Tabulka 3), který se volí podle argumentu funkce.



Obr.46: Diagram první části funkce OWPresentDevice

23 bytů verifikačních dat se získávají pomocí funkce bitacc. Jako vstupní informace pro funkci bitacc slouží ROM kód získaný z bufferu sériových čísel. Po vytvoření paketu se paket odesílá na sériovou linku (funkce WriteCOM).

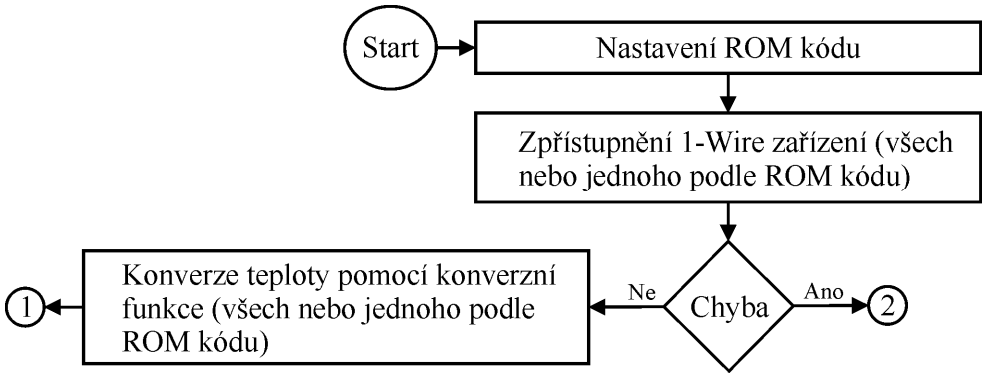


Obr.47: Diagram druhé části funkce OWPresentDevice

Druhá část funkce provádí rekonstrukci odpovědí (viz. Obrázek 47). Opět se používá funkce bitacc. Na základě návratové hodnoty této funkce se generuje vyhodnocovací proměnná tzv. goodbit. Na základě goodbitu se vyhodnocuje chyba. Pokud je goodbit správný (je získáno 8 bytů odpovídajících ROM kódu) ukončí se funkce a vrací se nulový příznak chyby (Return 0). Při detekci chyby v nějaké části funkce se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1).

6.2.12 Funkce OWReadTemp

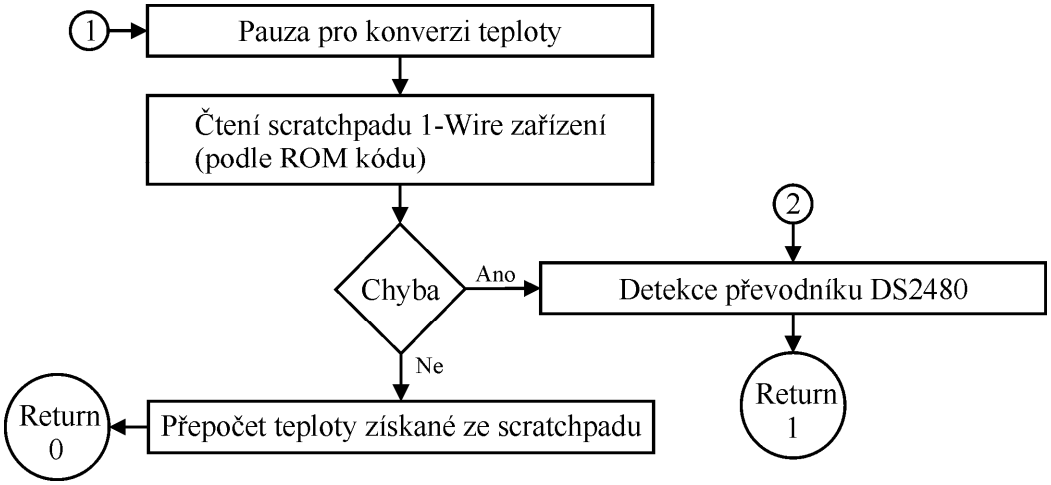
Tato funkce provádí konverzi a čtení teploty. Jako první se ve funkci volá funkce pro zpřístupnění jednoho nebo všech 1-Wire zařízení (viz. Obrázek 48). Pro zpřístupnění zařízení obsahuje funkce argument pole s ROM kódem a argument pro volbu přístupu do jednoho nebo všech zařízení. ROM kód argumentu funkce se pomocí funkce OWserialNum vkládá do bufferu sériových čísel. Na základě volby (pomocí příkazu if) přístupu se volá funkce OWMATCHROM nebo OWSkipMATCHROM.



Obr.48: Diagram první části funkce OWReadTemp

Po zpřístupnění 1-Wire zařízení se volá konverzní (viz. Obrázek 49) funkce (OWConvcert). Do argumentu této funkce se vkládá příkazový byte (44h) pro inicializaci (konverzi) teploty (viz. Tabulka 3). S tím souvisí nutnost zavedení zpoždění (viz. Kapitola 2.4). Realizuje se inkrementací pomocné

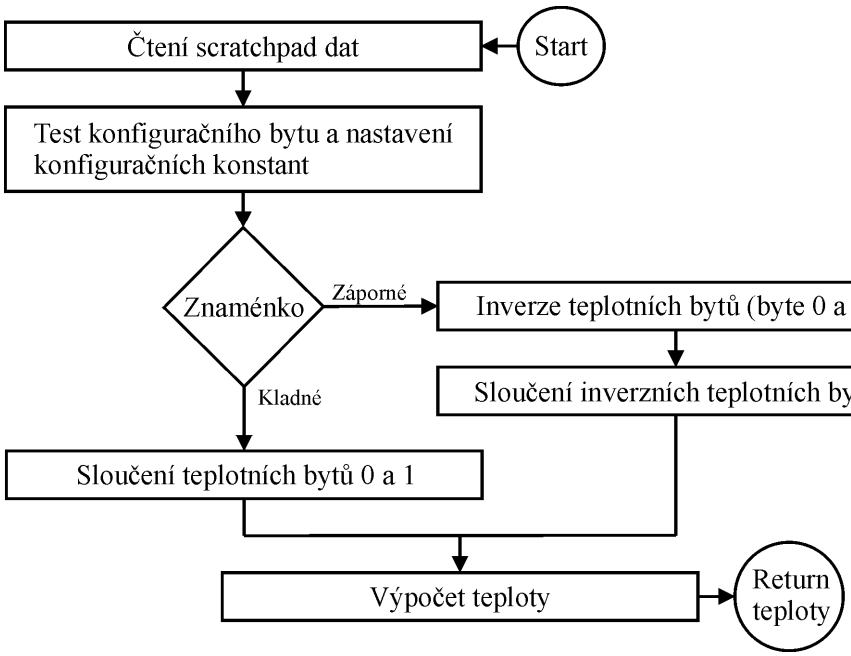
proměnné a srovnáním výsledku s argumentem funkce zpoždění. Po zpoždění se volá funkce pro čtení scratchpad paměti 1-Wire zařízení a navíc se nuluje pomocná proměnná pro realizaci zpoždění. Při detekci chyby v nějaké části funkce se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 1). Do argumentu funkce pro teplotu se vloží chybová konstanta (3276,7). Pokud, ale není detekována žádna chyba vykoná se funkce pro výpočet teploty (OWCalculationTemp). Výsledná teplota se uloží do argumentu funkce. Funkce se poté ukončí s nulovou návratovou hodnotou (Return 0).



Obr.49: Diagram druhé části funkce OWReadTemp

6.2.13 Funkce OWCalculationTemp

Tato funkce je určena pro výpočet teploty ze scratchpad dat. Vlastní přepočet vychází ze vzorců 1 a 2 uvedené v kapitole 5.1.



Obr.50: Diagram funkce OWCalculationTemp

Jako první se čte vstupní argument pole se scratchpad daty (viz. Obrázek 50). K tomuto účelu slouží funkce memcpy. Na základě čtvrtého bytu (konfigurační byte) scratchpad dat se získává konstanta pro posun (Drift) a konfigurační konstanta (ConvertConst) teploty (0,0625 => rozsah 12 bitů apod.). Po získání konstant se testuje horních 5 bitů prvního bytu. Na základě tohoto testu se získává informace o znaménku. Pokud je znaménko kladné vypočítává se teplota pomocí vzorce 1 z kapitoly 5.1.

```
TempWorld = ((UINT)(ReadBuffer[1])) << 8;                 //Posun bitů prvního bytu o 8 pozic
TempWorld = TempWorld + ((UINT)(ReadBuffer[0]));          //Sloučení prvního (MSB) a druhého (LSB) bytu
TempWorld = TempWorld >> Drift;                           //Posun bitů výsledného slova o Drift pozic
Temp = (REAL)(TempWorld * ConvertConst);                  //Výpočet Teploty
```

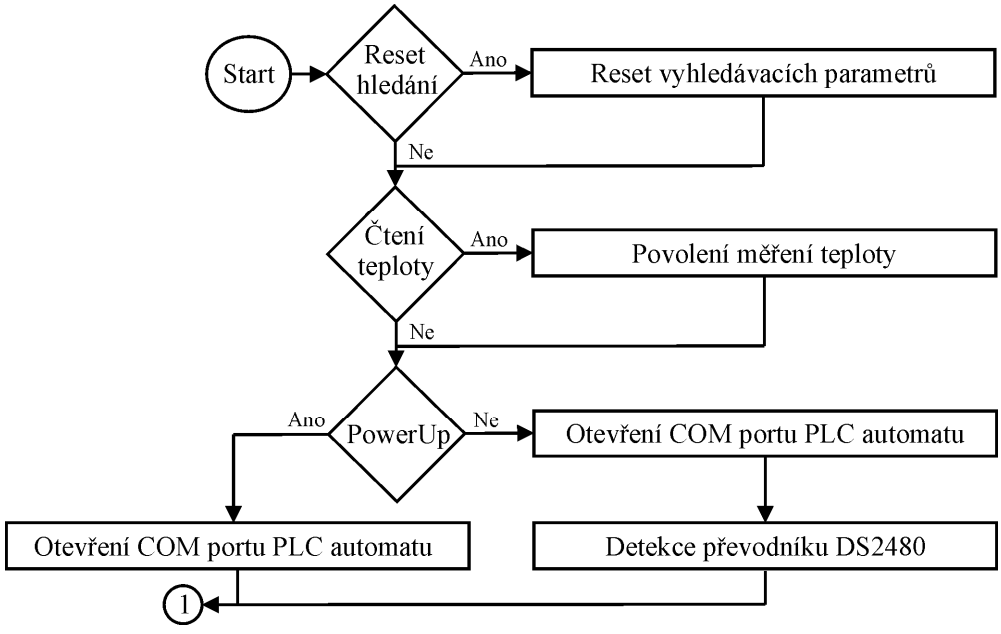
Pokud je znaménko záporné vypočítává se teplota pomocí vzorce 2 z kapitoly 5.1.

```
InvLSB = (UINT)(ReadBuffer[0] ^ 0xFF);                    //Inverze prvního (MSB) bytu
InvMSB = (UINT)(ReadBuffer[1] ^ 0xFF);                    //Inverze druhého (LSB) bytu
TempWorld = InvMSB << 8;                                  //Posun bitů MSB bytu o 8 pozic
TempWorld = TempWorld + InvLSB;                           //Sloučení MSB a LSB bytu
TempWorld = TempWorld >> Drift;                           //Posun bitů výsledného slova o Drift pozic
Temp = (REAL)(TempWorld * (ConvertConst * (-1)));         //Výpočet Teploty
```

Výsledná teplota se vrací v podobě návratové hodnoty (Return teploty).

6.2.14 Funkce OWMeasureTemp

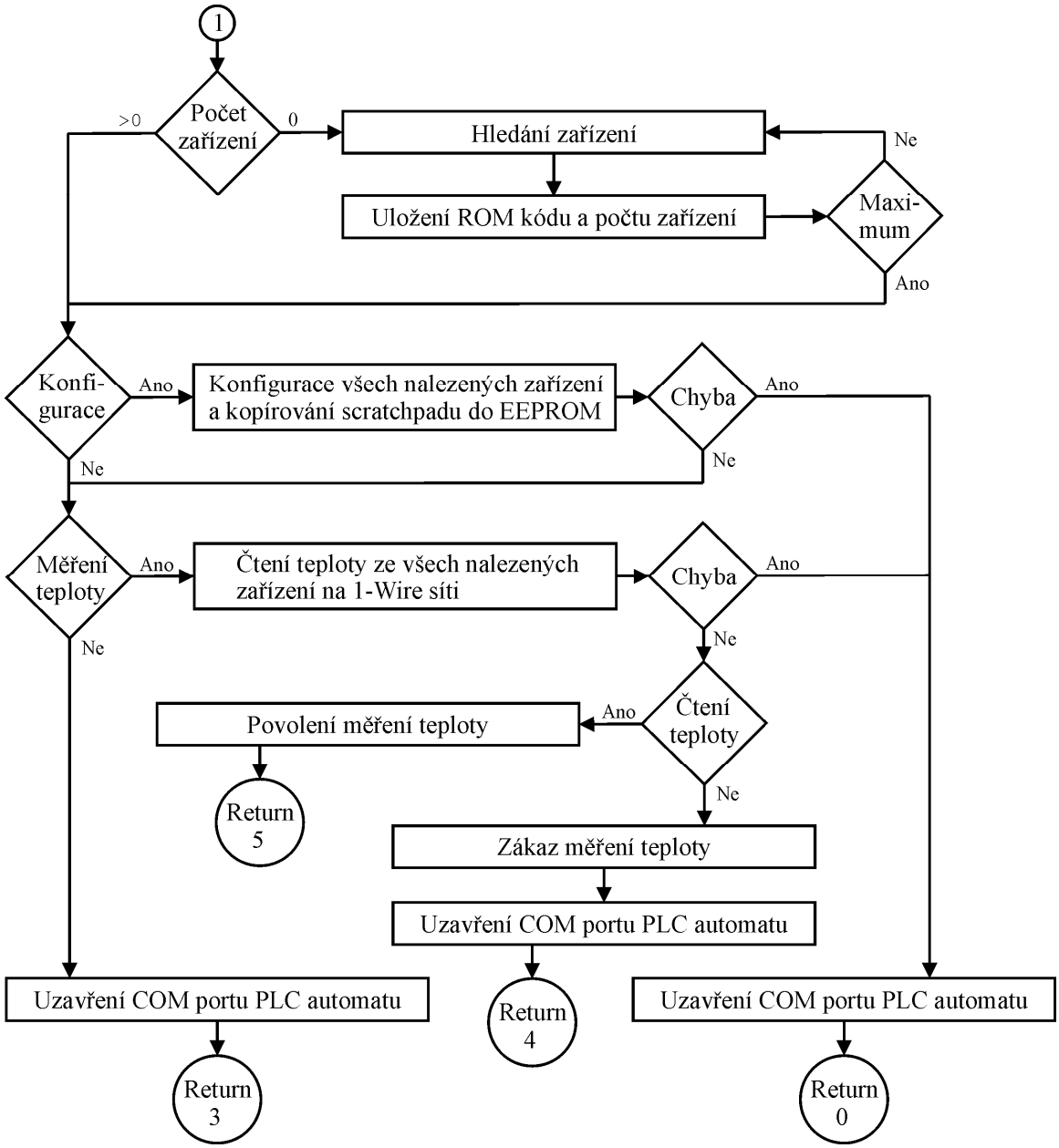
Jedná se o hlavní funkci, na jejímž základě se provádí celé měření a hledání zařízení 1-Wire sítě.



Obr.51: Diagram první části funkce OWMeasureTemp

Samotná funkce obsahuje několik kroků. V prvním kroku se testuje argument pro reset hledání a argument pro čtení teploty (viz. Obrázek 51). Pokud je argument resetu hledání v jedničce provede se vynulování vyhledávacích parametrů. Do proměnných „NumDev1 a NumDev2“ struktury RS232 se vloží nuly. Taktéž se vloží nuly i do argumentu udávající počet nalezených zařízení a argumentu resetu hledání. Pokud je argument čtení v jedničce, vloží se do pomocné proměnné povolení měření

jednička. V druhém kroku se testuje proměnná PowerUp, která zajišťuje otevření sériového portu a detekci převodníku DS2480 (ADA-101W). Dále proměnná PowerUp zajišťuje, že se detekce převodníku (funkce DS2480Detect) provede jen jednou nebo při resynchronizaci komunikace.



Obr.52: Diagram druhé části funkce OWMeasureTemp

Ve třetím kroku se testuje počet nalezených zařízení argumentu funkce (viz. Obrázek 52). Pokud je argument funkce nulový volá se funkce vyhledávání (OWFindDevices). Tato funkce se vykonává neustále dokola (současně se ukládá i ROM kód nalezeného zařízení do pomocného pole a do argumentu funkce), dokud není nalezeno maximum zařízení 1-Wire sítě. Maximum zařízení, které lze vyhledat jsou definovaný počtem zařízení na 1-Wire sítí a argumentem funkce. Pokud je nalezeno maximum zařízení uloží se jejich počet do argumentu počtu nalezených zařízení a proměnné NumDev1 struktury RS232.

Ve čtvrtém kroku se testuje argument funkce pro povolení konfigurace (jen pokud je počet nalezených zařízení větší než 0). Pokud je povolena konfigurace volá se funkce pro odeslání konfiguračních dat (OWWriteConfigAll) a funkce pro kopírování scratchpad dat do EEPROM paměti (OWCopyScratchpadAll). Obě funkce slouží pro současnou konfiguraci všech nalezených zařízení na 1-Wire síti. Konfigurační data z argumentu funkce se vloží do všech 1-Wire zařízení v jeden okamžik (současně).

V pátém kroku se testuje pomocná proměnná pro povolení měření (jen pokud je počet nalezených zařízení větší než 0). Pokud je měření povoleno volá se funkce čtení teploty (OWReadTemp). Tato funkce na základě ROM kódu z pomocného pole, proměnné NumDev1 a proměnné NumDev2 postupně čte teplotu a ukládá ji do argumentu funkce. Na konci měření se opět testuje argument povolení měření a vyhodnocuje se stejným způsobem jako v prvním kroku. Rozdílem je, že se na jeho základě ukončuje celá funkce a vrací se příznak chyby (Return 4 nebo Return 5).

Při detekci chyby v nějaké části funkce se volá funkce DS2480Detect pro obnovení synchronizace komunikace (vrací se příznak chyby => Return 0). Jsou-li všechny testované argumenty nulové, ale počet nalezených zařízení je větší než nula uzavře se sériový port, ukončí se funkce a vrací se příznak chyby (Return 3).

6.3 Funkce utilit

Jedná se o pomocné funkce, které doplňují ostatní funkce pomocnou vlastností. Mezi tyto funkce patří funkce uvedené v tabulce 9.

Funkce	Popis funkce
msDelay	Funkce pro realizaci krátkého zpoždění.
HexToChar	Funkce pro převod 8 bitového hexadecimálního čísla na dva znaky.
SetCRC8	Funkce pro vynulování výsledku výpočtu 8 bitového CRC kódu a inicializaci tabulky 8 bitových CRC hodnot převodníku DS2480.
DoCRC8	Funkce pro výpočet 8 bitových CRC hodnot.
ResetTableCRC8	Funkce obsahuje tabulku 8 bitových CRC hodnot převodníku DS2480.
ReadPowerUp	Funkce vrací hodnotu proměnné PowerUp (indikuje přítomnost převodníku ADA-101W)

Tab.9: Funkce pro obsluhu sériového portu PLC automatu

6.3.1 Funkce msDelay a ReadPowerUp

Obě funkce jsou velice jednoduché. Funkce ReadPowerUp vrací informaci o přítomnosti převodníku ADA-101W. K tomu to účelu slouží příkaz return.

Funkce msDelay realizuje krátké zpoždění. Pro realizaci se využívá cyklus for. Nevýhodou tohoto řešení je nutnost dávat pozor v jakém tasku se funkce nachází. Proto je využita pouze pro realizaci zpoždění v řádech jednotek milisekund.

6.3.2 Funkce HexToChar

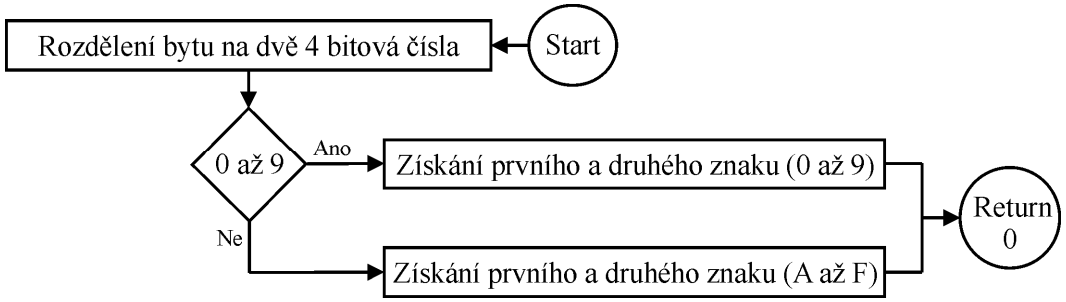
Tato funkce převádí 8 bitové číslo v hexadecimálním tvaru na dva znaky (viz. tabulka 10).

Hexadecimální tvar	První znak	Druhý znak
F1h	F	1
5Ah	5	A
BCh	B	C
64h	6	4

Tab.10: Příklad převodu

Převod je velice jednoduchý a má dvě části. V první části se převádí 8 bitové číslo na dvě 4 bitové čísla. První 4 bitové číslo se dostane posunem horních 4 bitů o 4 pozice směrem dolů (Hexadecimální tvar >> 4). Druhé 4 bitové číslo se dostane logickým násobením hexadecimálního tvaru s tzv. maskou (Hexadecimální tvar & 0x0F). Tím to násobením se vynulují horní 4 bity a zůstanou jen dolní 4 bity.

4 bitová hodnota odpovídá znakům => 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E a F. Proto se v druhé části testuje rozsah 0 až 9. Pokud je 4 bitové číslo v rozsahu 0 až 9 generují se znaky 0 až 9. Pokud je 4 bitové číslo mimo rozsah 0 až 9 generují se znaky A až F.



Obr.51: Diagram funkce HexToChar

6.3.3 Funkce pro výpočet 8 bitového CRC

Každé 1-Wire zařízení poskytuje pro kontrolu chyby CRC. V případě teplotního čidla DS18B20 se jedná o 8 bitovou hodnotu. Je součástí scratchpad paměti (viz. Obrázek 20). Jelikož je potřeba testovat přijímané informace z 1-Wire sítě musí Master zařízení (PLC Systém X20) přijímané informace přepočítávat. To znamená, že se vypočítává 8 bitový CRC kód stejným způsobem jako v zařízení na 1-Wire síti. K tomuto účelu slouží polynomiální vzorec:

$$CRC = x^8 + x^5 + x^4 + 1 \qquad (3)[3]$$

Výpočet 8 bitového CRC kódu zajišťují tři funkce (SetCRC8, DoCRC8 a ResetTableCRC8). Funkce SetCRC8 slouží pro vynulování proměnné s výsledkem 8 bitového CRC a inicializaci tabulky 8 bitových CRC hodnot. Funkce ResetTableCRC8 obsahuje tabulku 8 bitových CRC hodnot potřebných pro převodník DS2480.

```
DS2480CRCTable[ ] = {
      0, 94,188,226, 97, 63,221,131,194,156,126, 32,163,253, 31, 65,
    157,195, 33,127,252,162, 64, 30, 95,  1,227,189, 62, 96,130,220,
     35,125,159,193, 66, 28,254,160,225,191, 93,  3,128,222, 60, 98,
    190,224,  2, 92,223,129, 99, 61,124, 34,192,158, 29, 67,161,255,
     70, 24,250,164, 39,121,155,197,132,218, 56,102,229,187, 89,  7,
```

```
219,133,103, 57,186,228,  6, 88, 25, 71,165,251,120, 38,196,154,
101, 59,217,135,  4, 90,184,230,167,249, 27, 69,198,152,122, 36,
248,166, 68, 26,153,199, 37,123, 58,100,134,216, 91,  5,231,185,
140,210, 48,110,237,179, 81, 15, 78, 16,242,172, 47,113,147,205,
 17, 79,173,243,112, 46,204,146,211,141,111, 49,178,236, 14, 80,
175,241, 19, 77,206,144,114, 44,109, 51,209,143, 12, 82,176,238,
 50,108,142,208, 83, 13,239,177,240,174, 76, 18,145,207, 45,115,
202,148,118, 40,171,245, 23, 73,  8, 86,180,234,105, 55,213,139,
 87,  9,235,181, 54,104,138,212,149,203, 41,119,244,170, 72, 22,
233,183, 85, 11,136,214, 52,106, 43,117,151,201, 74, 20,246,168,
116, 42,200,150, 21, 75,169,247,182,232, 10, 84,215,137,107, 53};
```

Funkce DoCRC8 je hlavní funkce pro výpočet 8 bitového CRC. V podstatě se jedná o CRC generátor. CRC kód se vypočítává tak, že se výsledek výpočtu z předchozího děje sečte (XOR => exkluzivní logický součet) s přijatým bytem. Výsledek tohoto logického součtu ukazuje na pozici v poli CRC hodnot. Výsledkem výpočtu 8 bitového CRC kódu je hodnota z pole DS2480CRCTable uložená v proměnné UtilCRC8.

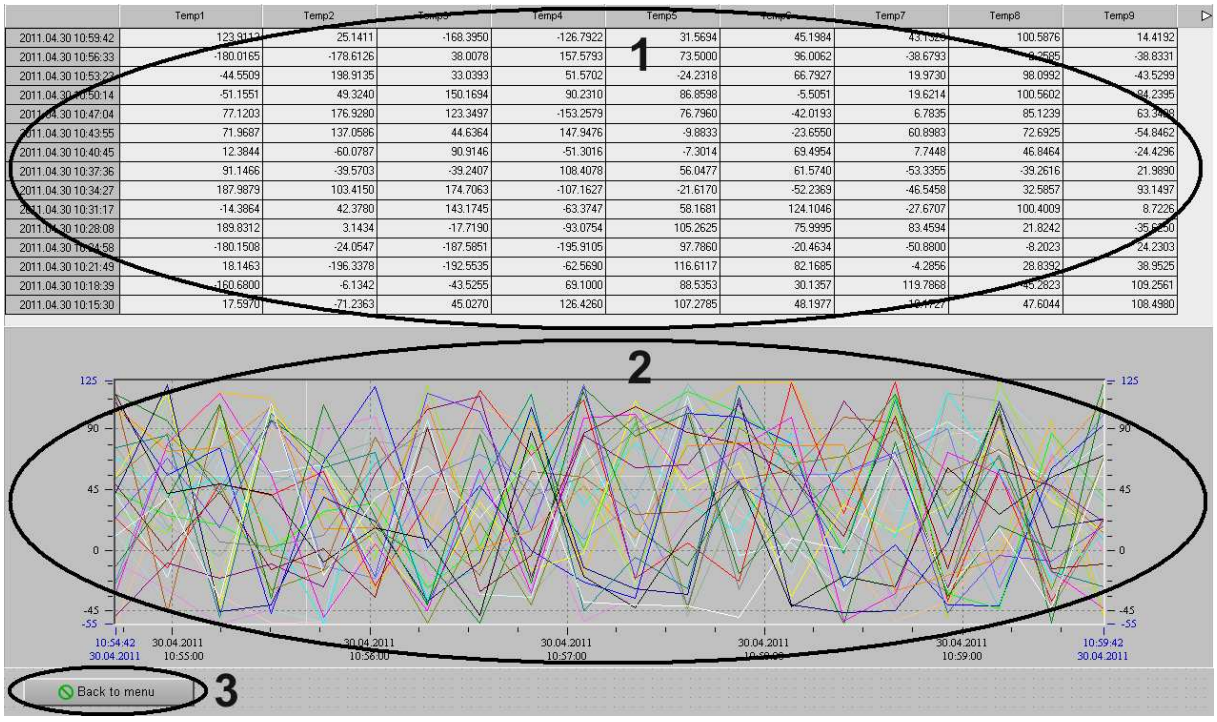
```
UtilCRC8 = DS2480CRCTable[(unsigned int) (UtilCRC8 ^ x)];
return UtilCRC8;
```

V případě výpočtu 8 bitového CRC kódu scratchpad paměti se tento děj provede 8 krát a nakonec se porovná s 9 hodnotou, která odpovídá CRC kódu generovaného 1-Wire zařízením. Pokud jsou tyto hodnoty shodné, přijaté data jsou v pořádku. V opačném případě jsou data chybná.

7 Implementace knihovny

7.1 Vizualizační aplikace pro nastavování parametrů měření

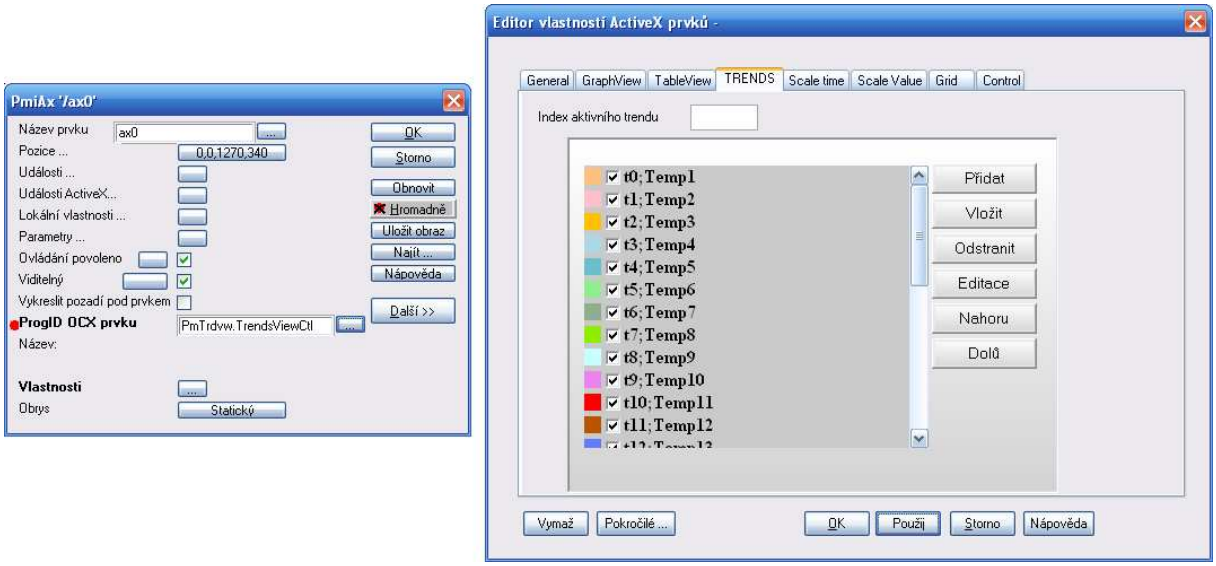
Realizovaná vizualizační aplikace je vytvořena pomocí vývojového prostředí Promotic. Tato aplikace má dvě verze. První verze je určena pro testování v laboratorních podmínkách. Druhá verze je určena pro provoz v reálném prostředí vrtů. Obě verze jsou v základu stejné. Rozdílem je, že druhá aplikace obsahuje ještě jeden sloupec pro indikaci umístění čidel ve vrtu. Tato informace je neměnná a není nijak ovlivňována. Druhým rozdílem obou verzí je nastavení zdroje OPC dat. V obou případech se pro vyčítání OPC dat používá PVI aplikace, která obsahuje OPC server (B&R.PviOPC.2 nebo BR.OPC.Server_3.0_090512_V1.14.1).V první verzi se OPC server konfiguruje pomocí konfiguračního souboru vytvořeného v testovací aplikaci PLC automatu (viz. Kapitola 7.2). Ve druhé verzi se OPC server konfiguruje pomocí aplikace PVI OPC konfigurátoru (viz. Kapitola 7.3). Samotná vizualizace se skládá ze dvou oken. První okno „Hlavní okno“ obsahuje nastavovací prvky (viz. Část 2 a 3 na obrázku 54), vizualizaci komunikace (viz. Část 1 na obrázku 54) a tabulku dat (viz. Část 4, 5 a 6 na obrázku 54). Druhé okno „Okno reálného trendu“ obsahuje dva prvky reálného trendu (grafický „Část 2 na obrázku 52“ a tabulkový „Část 1 na obrázku 52“) a tlačítko pro návrat do hlavního okna (viz. Část 3 na obrázku 52). Hlavní část vizualizace tvoří OPC. OPC se konfiguruje pomocí OPC klienta, ve kterém je nastavený zdroj OPC dat. OPC data pak obsahuje tabulkový blok OPC data.



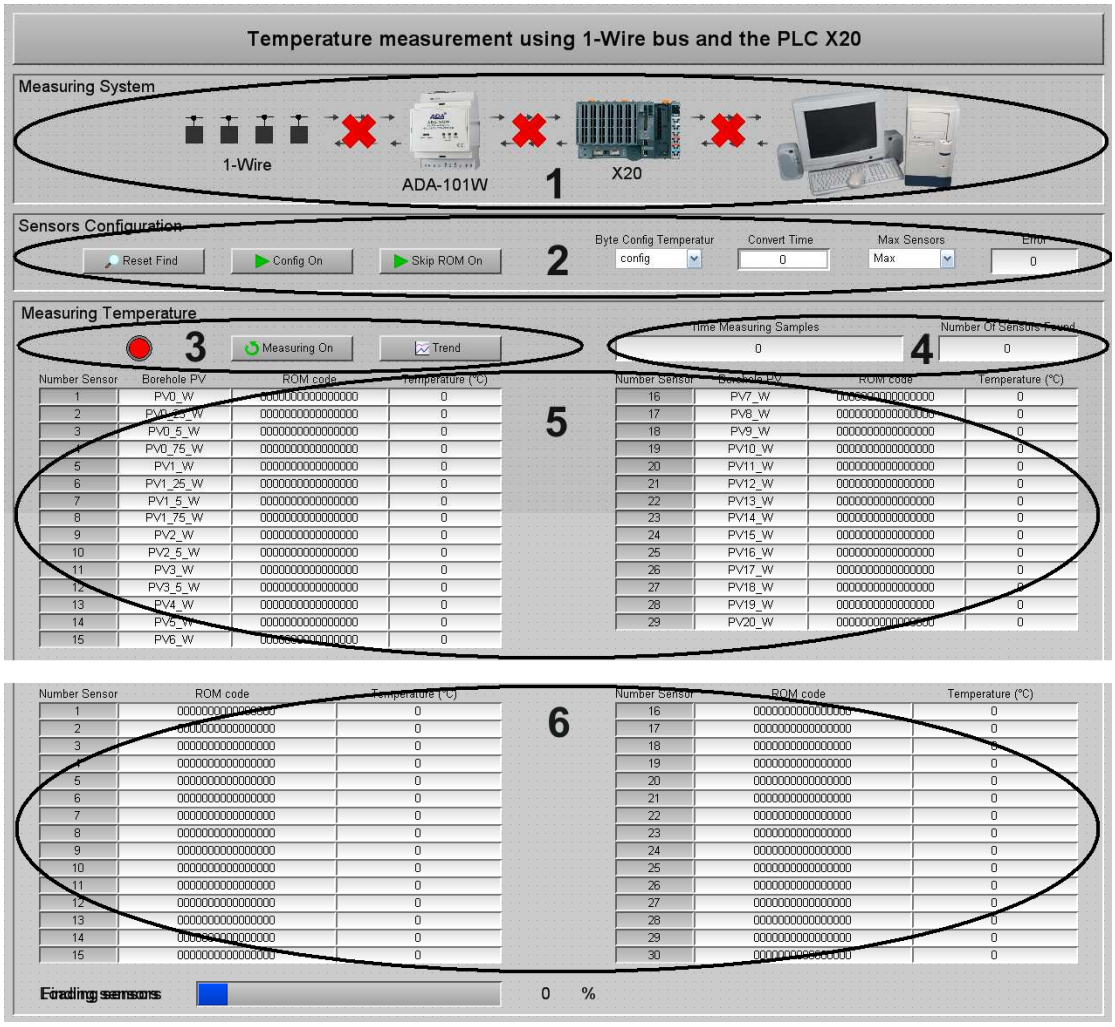
Obr.52: Screen okna reálného trendu

Reálný trend je tvořen ActiveX prvkem PmiAx tzv. TrendsView. Tento prvek obsahuje pouze data z tabulky trendů, jež jsou odvozeny z OPC dat (viz. Obrázek 53). Samotné tlačítko obsahuje pouze jednoduchý skript realizující uzavření okna trendů a otevření hlavního okna.

```
Pm("/Historie").Close
Pm("/MainPanel").Open
```



Obr.53: Nastavení reálného trendu



Obr.54: Screen hlavního okna

Vizualizace komunikace (viz. Část 1 na obrázku 54) je tvořena několika obrázky, které se zobrazují podle OPC proměnné. Tato OPC proměnná obsahuje návratovou hodnotu funkce měření teploty (OWMeasureTemp). U obrázku červeného křížku se navíc realizuje blikání. Blikání je realizováno pomocí Timeru a průsvitnosti objektu. Timer se vykonává co 0,5 sekundy a obsahuje jednoduchý skript pro nastavení úrovně průsvitnosti. Skript pouze přiřazuje konkrétní hodnotu, na jejímž základě se objekt zobrazí nebo nezobrazí.

```
If Pm("Efect").Value = 100 then
    Pm("Efect").Value = 0
Else
    Pm("Efect").Value = 100
End If
```

Kromě vizualizace komunikace aplikace obsahuje i prvek bar grafu tzv. PmiBar. Tento bar graf se zobrazuje na základě čtení dat teploty nebo při hledání nových zařízení. Bar graf obsahuje tři proměnné. Jedna proměnná slouží pro zobrazení tohoto grafu. Další dvě slouží pro nastavení hraničních hodnot bar grafu. Tyto tři proměnné se nastavují pomocí skriptu. Tento skript je opět obsažen v Timeru. Tento Timer a vykonává co 0,1 sekundu. Na základě tohoto skriptu se také vypočítává procentuální hodnota bar grafu. Všechny tři proměnné se získávají z OPC proměnných.

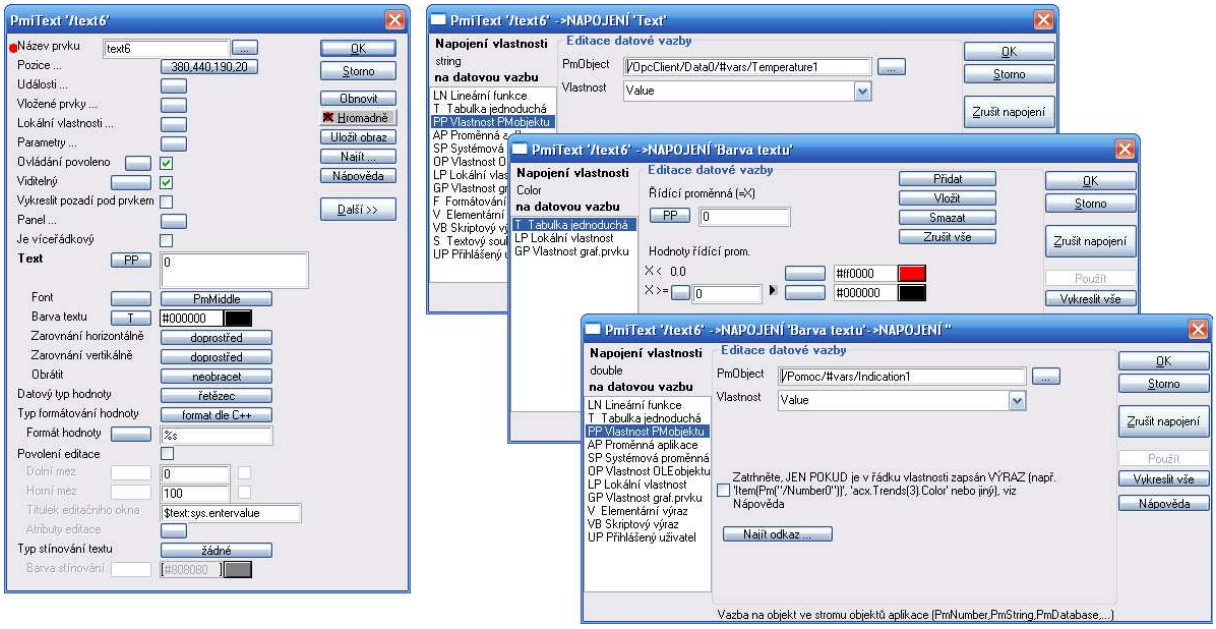
```
If OPCMeasureTemp = 1 then
    Pm("IndicationComunication").Value = 0
    If OPCNumDevices = 0 then
        Pm("LoadSensor").Value = 0
        Pm("FindSensor").Value = 1
        Pm("Number").Value = (OPCNumDev1 / (OPCMaxDevice / 100)) * 100
        Pm("Procento").Value = (Pm("Number").Value / 100)
    Else
        Pm("LoadSensor").Value = 1
        Pm("FindSensor").Value = 0
        Pm("Number").Value = (OPCNumDev2 / (OPCNumDev1 / 100)) * 100
        Pm("Procento").Value = (Pm("Number").Value / 100)
    End If
Else
    Pm("IndicationComunication").Value = 1
    Pm("LoadSensor").Value = 0
    Pm("FindSensor").Value = 0
End If
```

Hlavní okno obsahuje několik typů nastavovacích prvků (viz. Část 2 a 3 na obrázku 54). Jedná se o tlačítka (tzv. PmiButton), combo boxy (tzv. PmiWCombo) a text boxy (tzv. PmiText). Všechny tyto bloky obsahují pro nastavení konkrétní proměnnou z OPC klienta. Jedním rozdílem je tlačítko Trend. Toto tlačítko obsahuje skript v podobě jednoho řádku. Pomocí tohoto skriptu se zobrazí okno trendů „Okno reálného trendu“.

```
Pm("/Historie").Open
```

Poslední vizualizační částí je oblast dat (viz. Část 4, 5 a 6 na obrázku 54). Měřená data se zobrazují do tabulky vytvořené pomocí text boxů tzv. PmiText. Zhotovená tabulka je dvojí podle

aplikace, ve které je použita. Samotná funkčnost a struktura je u obou tabulek stejná. Text boxy pro zobrazení ROM kódu, času měřeného vzorku a počtu nalezených zařízení obsahují pouze konkrétní proměnné z OPC klienta. Text boxy pro zobrazení informace o teplotě obsahují konkrétní proměnné OPC klienta s hodnotou teploty a navíc proměnnou, která mění barvu textu v závislosti na změně hodnoty teploty (viz. Obrázek 55).



Obr.55: Příklad nastavování text boxu

Proměnná, která generuje změnu barvy, se realizuje pomocí skriptu dvou Timerů. První Timer se vykonává co 1 sekundu a obsahuje skript pro přiřazení OPC proměnné do pomocné proměnné.

```
Pm("/Pomoc").Item("Temperature1a").Value = Pm("/OpcClient/Data0").Item("Temperature1").Value
.
.
Pm("/Pomoc").Item("Temperature30a").Value = Pm("/OpcClient/Data0").Item("Temperature30").Value
```

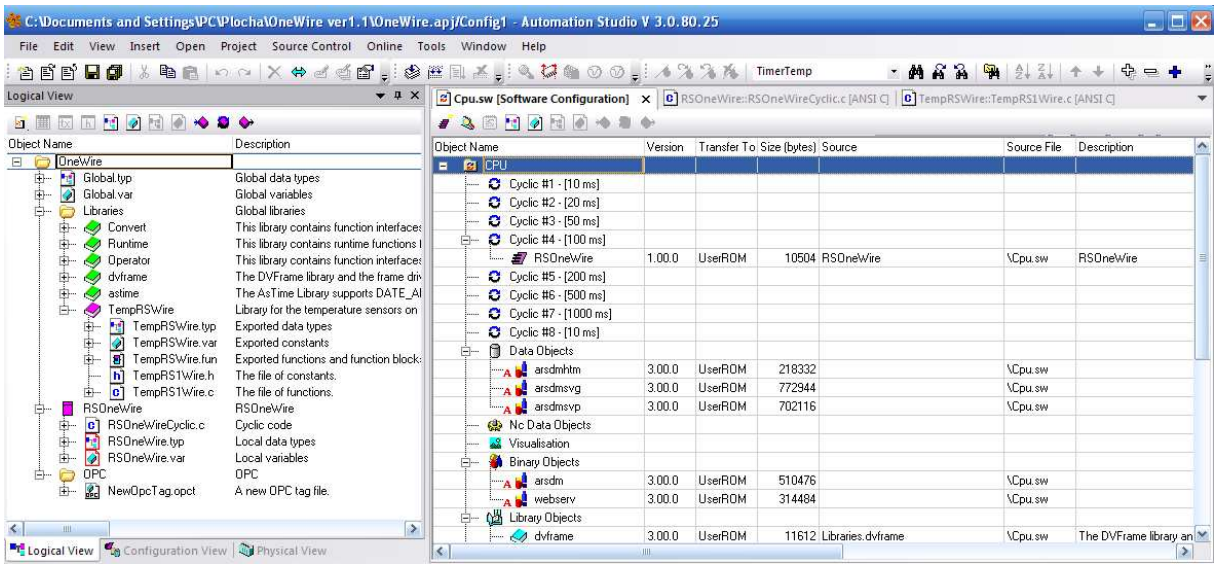
Druhý Timer se vykonává co 0,5 sekund. Jedná se o již zmíněný Timer pro generování proměnné vizualizace komunikace. Tento Timer ještě obsahuje skript pro vyhodnocení rovnosti aktuální teploty a dříve zjištěné teploty. Pokud se teploty nerovnají, generuje se hodnota proměnné pro změnu barvy textu. V opačném případě je tato proměnná nulová a barva se nemění.

```
If Pm("/Pomoc").Item("Temperature1a").Value = Pm("/OpcClient/Data0").Item("Temperature1").Value then
    Pm("/Pomoc").Item("Indication1").Value = 0
Else
    Pm("/Pomoc").Item("Indication1").Value = -1
End If
```

Příklady výsledných vizualizačních aplikací jsou uvedený v následujících kapitolách na obrázku 58 a na obrázku 62.

7.2 Testování knihovny v laboratoři

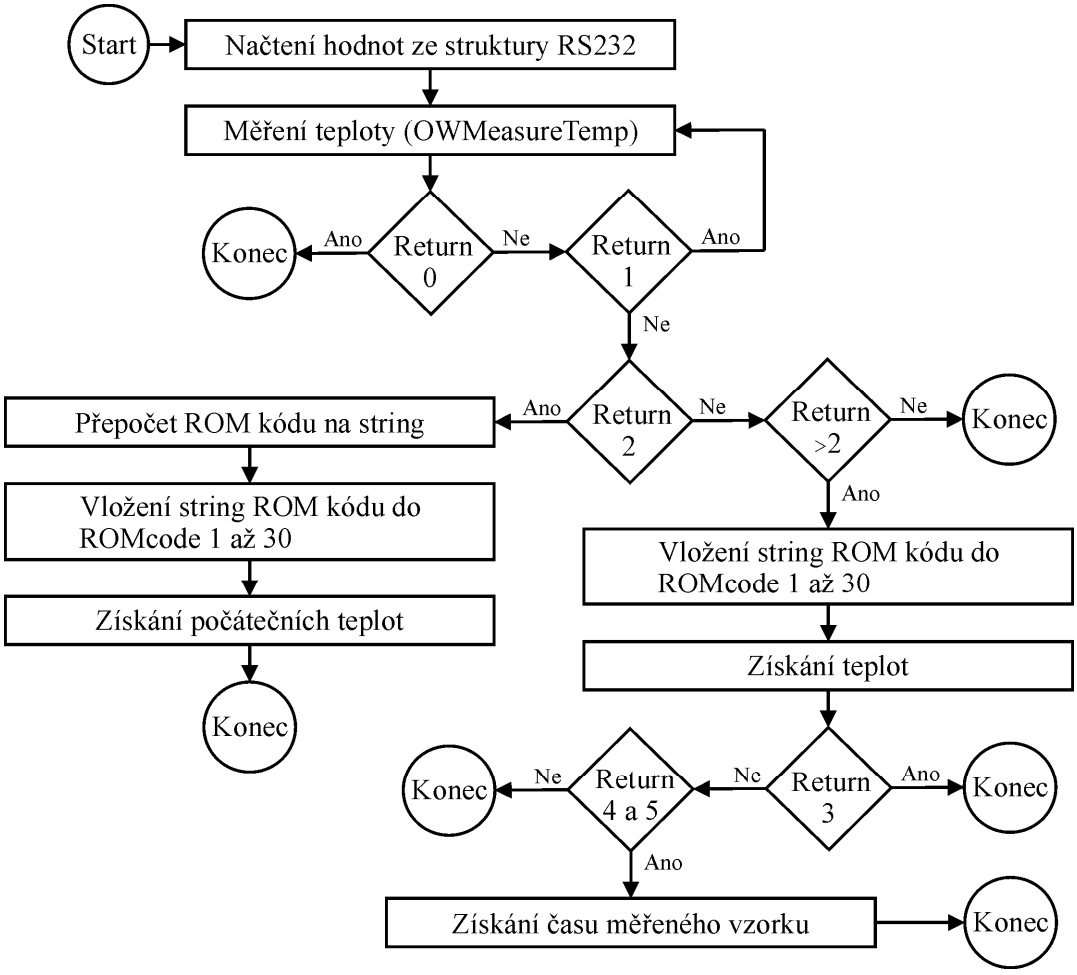
Pro laboratorní účely je vytvořen model navržené měřicí struktury (viz. Obrázek 12). Tento model obsahuje pouze komponenty pro realizaci měření na síti 1-Wire. Model tedy obsahuje pouze PLC automat řady Systém X20, převodník ADA-101W a kabel s čidly DS18B20.



Obr.56: Screen testovací aplikace v prostředí Automation Studia

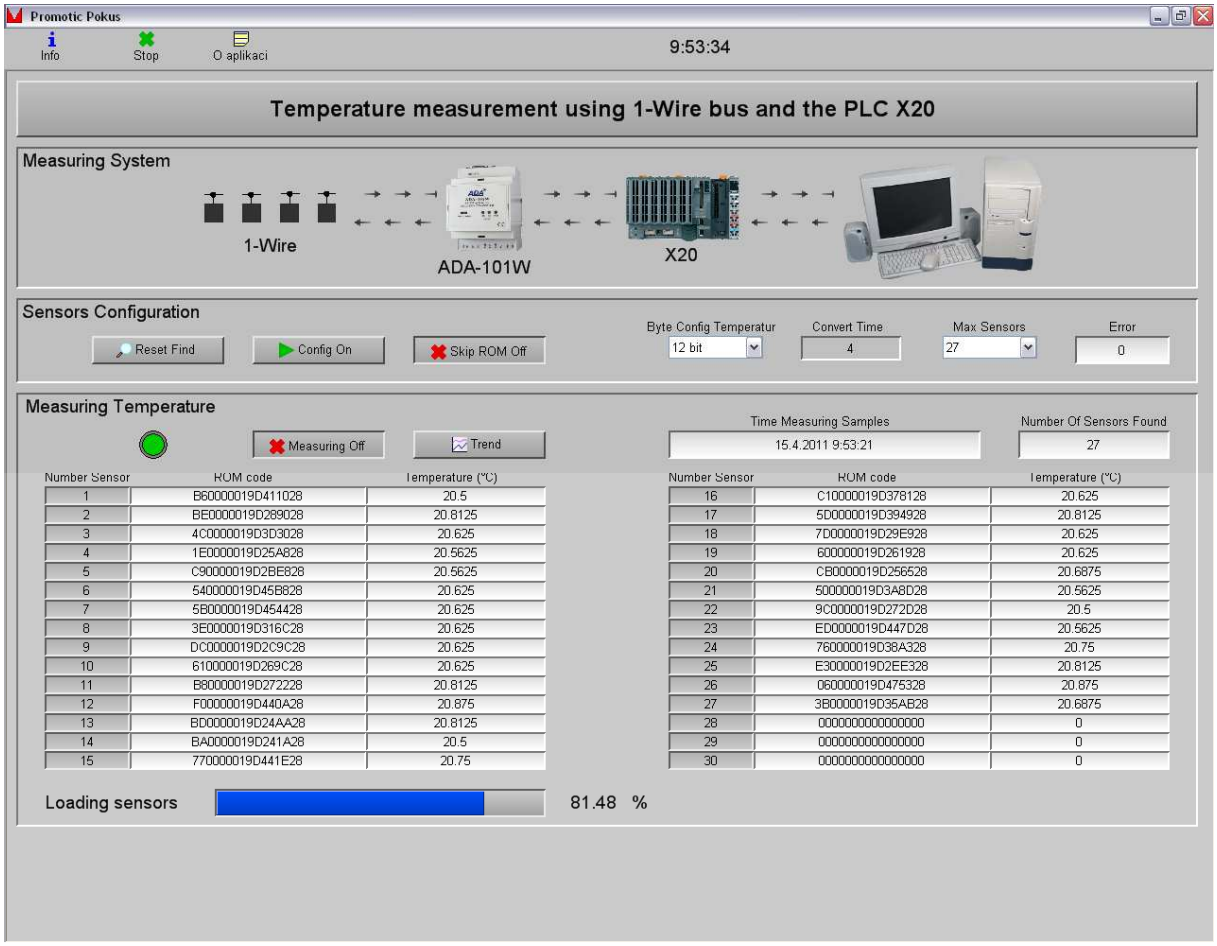
Součástí modelu je i nově vytvořená aplikace. Tato aplikace je vytvořena pomocí vývojového prostředí B&R Automation Studio. I přes možnost volby až z osmi programovacích jazyků (ANSI C, Automation Basic, Ladder Diagram, Strukturovaný text, atd.) je aplikace psána pomocí programovacího jazyka ANSI C. Aplikace obsahuje pouze jeden taskový cyklus, který je nastaven tak, aby se vykonával co 100 milisekund. Jedná se o počáteční nastavení při vytváření tasku. Task je možné umístit i do jiných časových režimů (viz. Obrázek 56), ale tím by se zpomalilo vykonávání funkcí knihovny a následně by se zpomalilo i měření. Tento cyklus obsahuje implementaci realizované knihovny. Ta se zde přidává stejným způsobem jako jiné dostupné knihovny. Aplikace kromě realizované knihovny obsahuje i knihovny DVFrame, ASTime apod. Součástí aplikace je v neposlední řadě tzv. OPC tag. Do tohoto tagu jsou přidány všechny globální proměnné používané v aplikaci. PLC automat na základě tohoto tagu generuje OPC data. Tyto data jsou potřebné pro vizualizační aplikaci v Promoticu. Vizualizace pomocí OPC dat komunikuje s PLC automatem.

Samotná aplikace je velice jednoduchá. Skládá se ze dvou částí. V první části se provádí inicializace veškerých proměnných. Tato část se vykonává pouze po restartu PLC automatu. Druhou část tvoří samotná aplikace (viz. Obrázek 57). Základ tvoří proměnné s daty získanými ze struktury RS232 realizované knihovny a funkce pro měření teploty (OWMeasureTemp) 1-Wire zařízení. Argumenty funkce tvoří globální proměnné, které jsou přidány i do OPC tagu. Pomocí zmíněných proměnných se provádí nastavování a čtení informací získaných měřením. Důležitou částí funkce pro měření teploty jsou chybové návratové hodnoty. Pomocí těchto chybových hodnot se určuje stav měření.

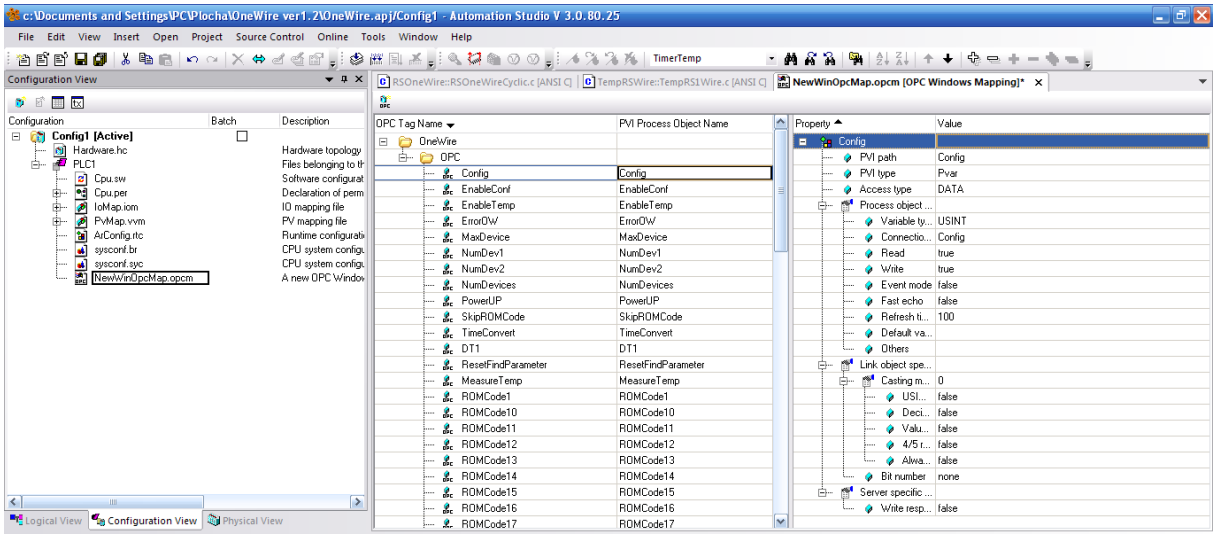


Obr.57: Diagram testovací aplikace

Je li návratová hodnota rovna nule (Return 0) nastala ve funkci chyba a aplikace tasku se ukončí. V případě, že je návratová hodnota rovna jedné (Return 1) funkce měření se opakuje. Je li návratová hodnota rovna dvěma (Return 2) nalezl se maximální počet 1-Wire zařízení. Na základě této návratové hodnoty se volá funkce HexToChar. Funkce HexToChar přepočítává ROM kód na formát string, pro lepší zpracovávaní. Dále se zde nastavují počáteční hodnoty teplot a jednotlivé ROM kódy se vkládají do OPC proměnných. Pokud je návratová hodnota větší než dvě došlo ke změření teploty. ROM kódy se opět vloží do OPC proměnných a taktéž se do OPC proměnných vloží získané teploty. Po tomto vkládání se opět testuje návratová proměnná. Pokud je rovna třem (Return 3) aplikace tasku se ukončí. Pokud je návratová hodnota rovna čtyřem nebo pěti (Return 4 a 5) přečte se globální čas PLC automatu a aplikace tasku se ukončí. Za tímto účelem slouží knihovna ASTime. Tento čas reprezentuje dobu měřeného vzorku.



Obr.58: Příklad obrazovky testovací vizualizace při činnosti



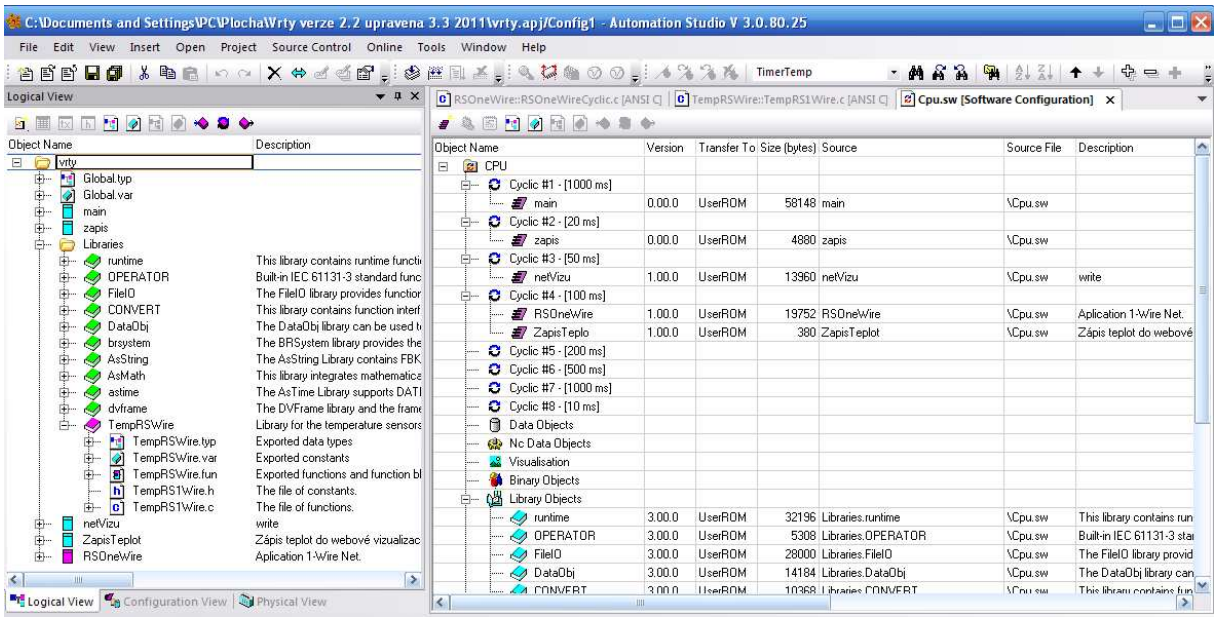
Obr.59: Nastavení OPC tagů

Na Obrázku 58 je vizualizační aplikace pro měření teploty na 1-Wire síti. Vizualizace umožňuje zobrazení měřených dat a konfiguraci proměnných testovací aplikace přes OPC. Pro propojení testovací aplikace PLC automatu a vizualizace v Promoticu se v tomto případě používá OPC server (BR.OPC.Server_3.0_090512_V1.14.1). OPC server je součástí PVI. Pro nastavení OPC serveru se

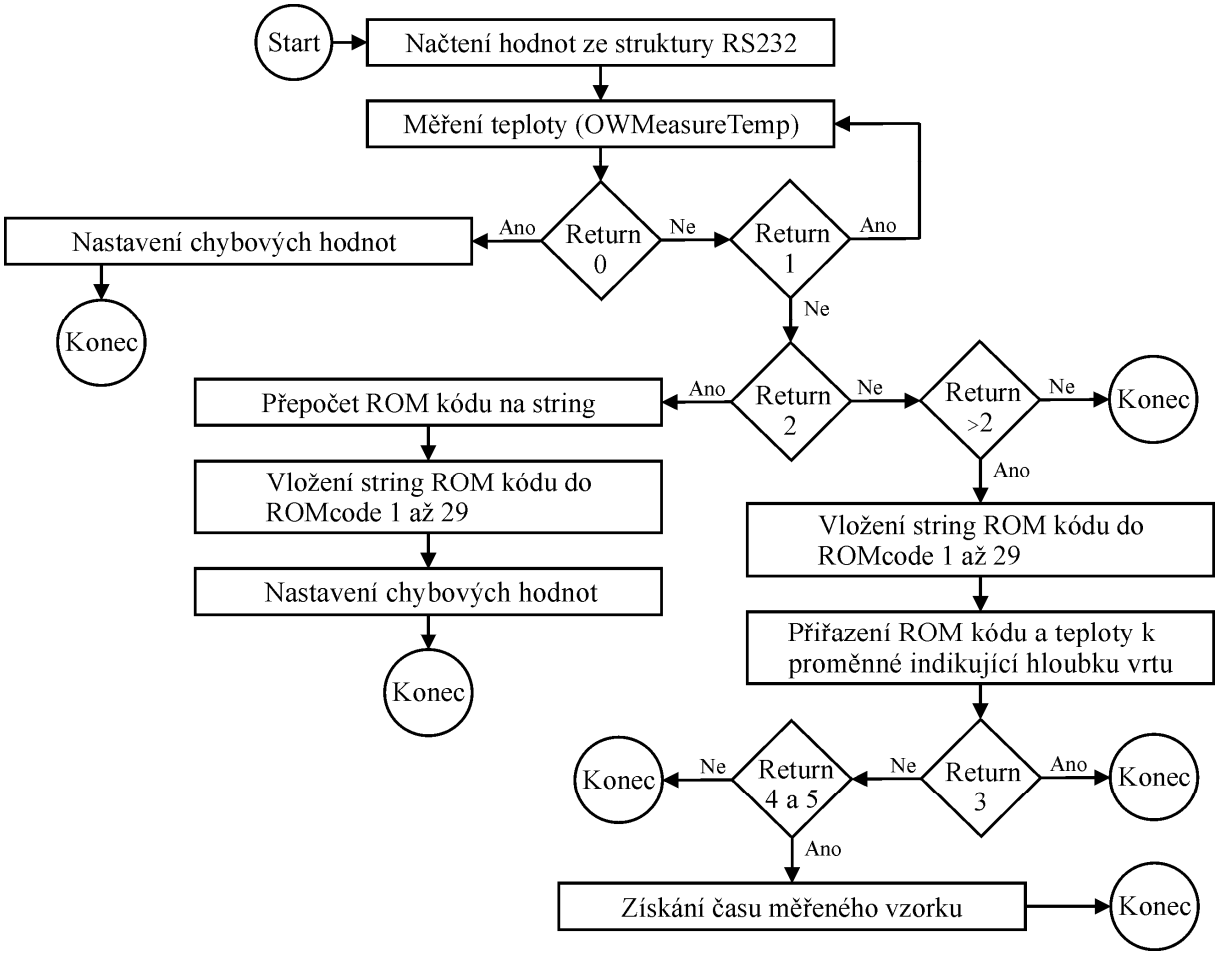
využívá konfigurační soubor PVI aplikace (BR.OPC.Server.ini). V tomto souboru se nastavuje název a cesta ke konfiguračnímu souboru OPC (NewWinOpcMap.opcs) testovací aplikace. OPC konfigurační soubor (NewWinOpcMap.opcs) testovací aplikace se generuje pomocí prostředí Automation studia na základě OPC tagů (viz. Obrázek 59).

7.3 Testování knihovny v reálném provozu

Jako reálný provoz se využívá navržený systém s 1-Wire sítí (viz. Obrázek 12). Pro ovládání tohoto systému se využívá stávající aplikace vytvořená pomocí vývojového prostředí Automation studio. Aby bylo možné realizovanou knihovnu otestovat je stávající aplikace obohacená o realizovanou knihovnu a task pro její obsluhu (viz. Obrázek 60). Task je nastaven ve 100 milisekundovém časovém režimu což znamená, že se task obnovuje co 100 milisekund. Volený časový režim je závislý na čase čtení dat ze sériového portu, proto je možné zvolit i nižší časovou třídu, ale optimální je 100 milisekundový. V případě vyšších časových režimů (500 a 1000 milisekund) by se stejně jako u testovací aplikace zpomalilo měření, jelikož by se zpomalilo čtení dat ze sériové linky.



Obr. 60: Screen aplikace pro reálný provoz v prostředí Automation Studia



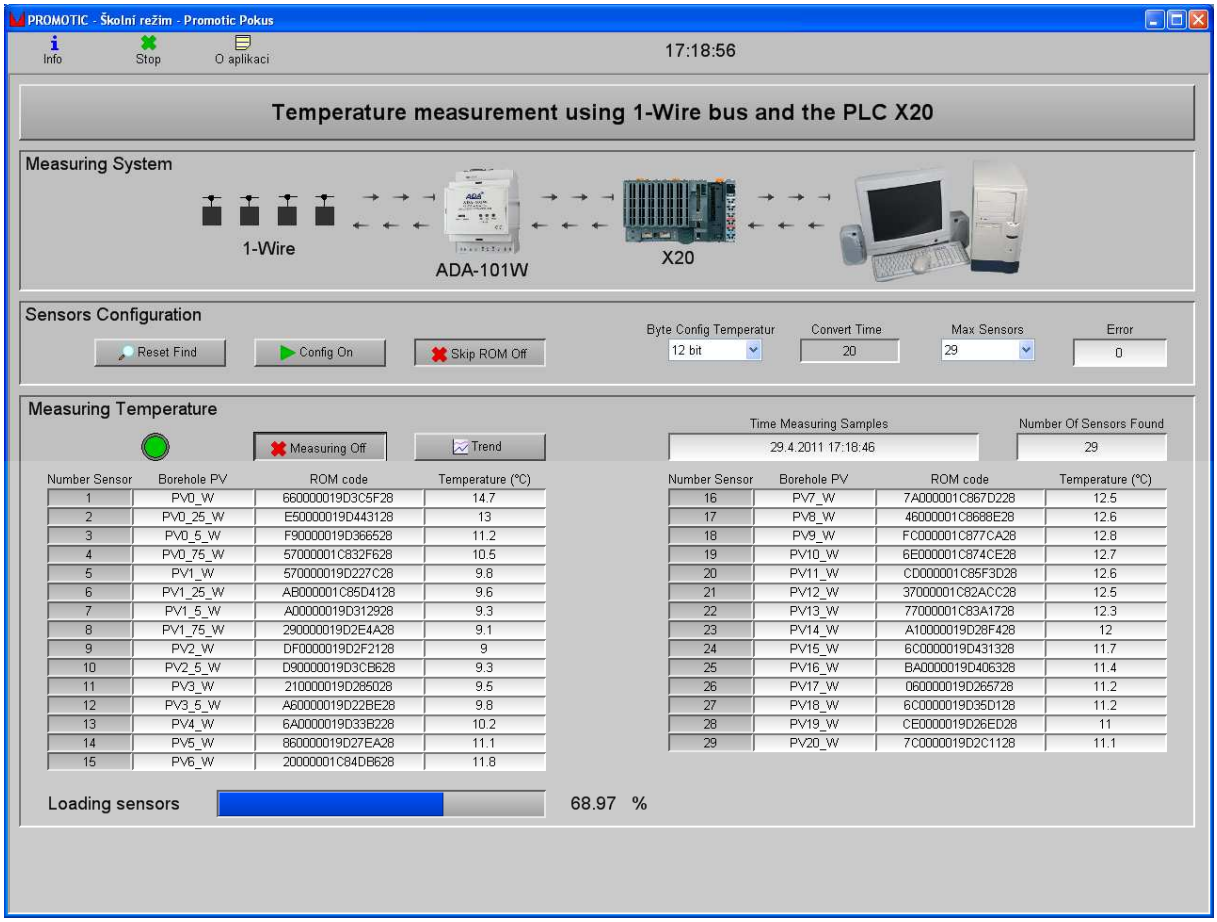
Obr.61: Diagram aplikace pro reálný provoz

Aplikace tasku je psána stejně jako testovací aplikace pomocí programovacího jazyka ANSI C. Tato aplikace má v podstatě stejnou strukturu jako testovací aplikace (viz. Obrázek 61 a 57). Rozdíl nastává v případě vyhodnocování teplot a návratových hodnot funkce pro měření teploty. V případě nulové (Return 0) návratové hodnoty se nastavují proměnné, indikující umístění čidla (hloubka vrtu – viz. Obrázek 2), na chybové hodnoty. Chybové hodnoty se nastavují i v případě návratové hodnoty rovné dvěma (Return 2). Co se týče vyhodnocení teplot, nastává v případě návratových hodnot vyšších jak dvě (Return >2). Pro vyhodnocení teplot je vytvořen speciální algoritmus. Tento algoritmus na základě požadovaného ROM kódu přiřazuje tepoty do proměnných indikující polohu čidla.

```
if(strncmp((char*)TableROMCode[0], (char*)StringROMCode[j], 16) == 0){
        PV0_W = (INT)(Temperature[j] * 10);
        for(k = 0; k < 16; k++){
                ROMCode1[k] = StringROMCode[j][k];
        }
}
```

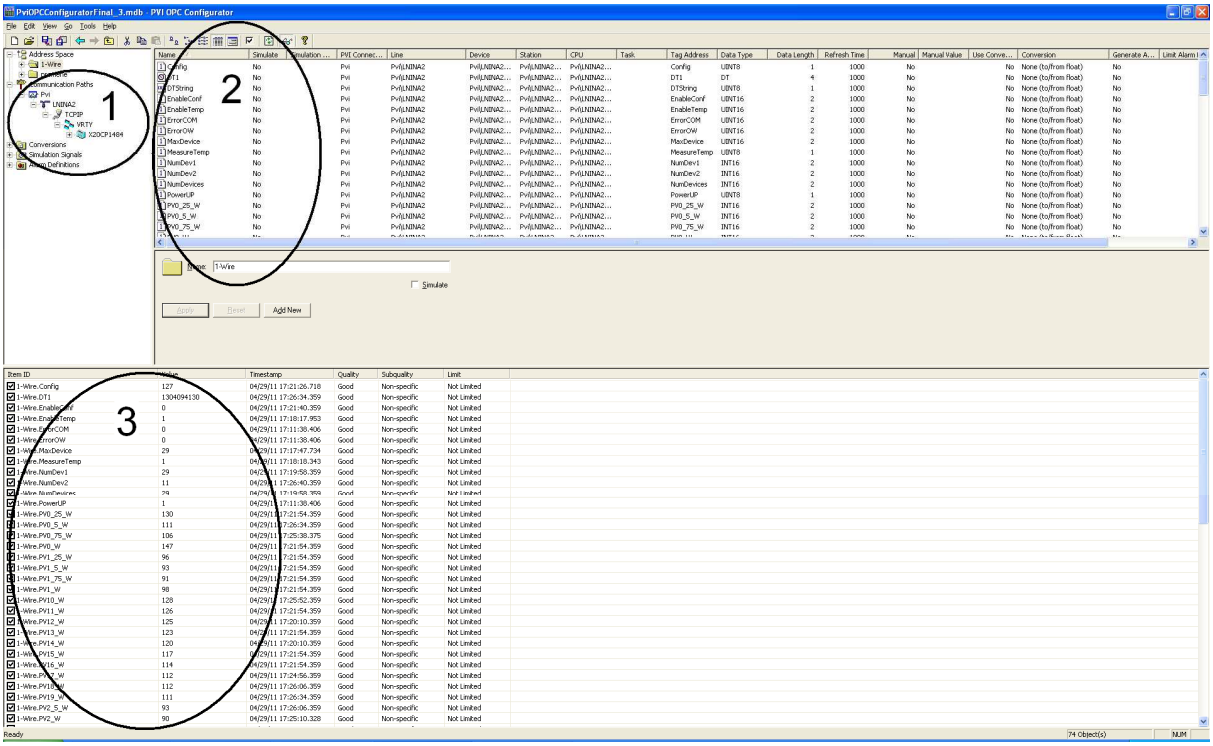
ROM kód se vyhodnocuje klasickým srovnáním ROM kódu vyhledaného zařízení a předem zjištěného ROM kódu. K porovnávání se používá funkce strncmp. Pokud jsou ROM kódy shodné, vrací se nulová hodnota. V opačném případě se vrací hodnota odpovídající pozici místa rozdílné hodnoty. Při nulové návratové hodnotě se násobí teplota 10 (vyhodnocení na jedno desetinné místo) a převádí se na celočíselnou hodnotu. Výsledná informace o teplotě se ukládá do proměnné, indikující

pozici čidla ve vrtu (PV0_W = 0m, PW1_W = 1m apod.). Současně se ukládá ROM adresa do proměnné ROM kódu. Jelikož je ROM kód pole hodnot realizuje se ukládání pomocí funkce for.



Obr.62: Příklad obrazovky vizualizace reálného provozu při činnosti

Na Obrázku 62 je upravená vizualizační aplikace pro měření teploty na 1-Wire síti. Vizualizace umožňuje zobrazení měřených dat jednotlivých čidel pozorovacího vrtu a konfiguraci proměnných nastavující parametry měření. Pro propojení vizualizace a aplikace tasku je v tomto případě použita aplikace PVI manažeru a aplikace PVI OPC konfigurátoru. PLC automat pomocí ethernnetu komunikuje s PVI manažerem. PVI OPC konfigurátor převádí data PVI manažeru na data OPC serveru (B&R.PviOPC.2). Z hlediska konfigurace OPC serveru se v PVI OPC konfigurátoru nejprve vytváří cesta k proměnným PLC automatu (viz. Část 1 na obrázku 63). Na základě této cesty se generují proměnné (viz. Část 2 a 3 na obrázku 63) OPC serveru. Získané OPC proměnné se přiřazují do vizualizační aplikace Promoticu.



Obr.63: Nastavení PVI OPC konfigurátoru

8 Závěr

Cílem této diplomové práce bylo rozšířit stávající měřící systém o aplikaci měření teploty pomocí 1-Wire zařízení. Současně bylo cílem této práce zhotovit knihovnu pro realizaci komunikace mezi PLC automatem řady X20 a mikroLAN sítí typu 1-Wire. Za tímto účelem byl stávající systém obohacen o převodník ADA-101W a pro PLC automat byla vytvořena aplikace s realizovanou knihovnou. Na základě navrženého rozšíření byl vytvořen testovací model v laboratoři. Tento model byl vytvořen za účelem testování realizované knihovny.

Jelikož se při vývoji knihovny vyskytovala řada problémů (od verze vývojového prostředí až po problémy s funkčností sériového portu PLC automatu) byl vytvořen samostatný program pro komunikaci převodníku ADA-101W a stolního počítače. Výsledná aplikace je flexibilní a je možné ji použít ve výuce. Na základě této aplikace a vyřešených problémů s komunikací po sériovém portu se zhotovila vlastní knihovna. Aby byla flexibilní byla rozdělena do řady podfunkcí. Pomocí vytvořených funkcí je možné realizovat řadu operací od vyhledávání zařízení až po realizaci měření teploty. Jelikož jsou 1-Wire zařízení hojně rozšířeny i do jiných oblastí měření (měření vlhkosti, tlaku apod.) je možné knihovnu dále rozšiřovat.

Co se týče aplikace PLC automatu byla stávající aplikace rozšířena o realizovanou knihovnu a task s procedurou měření teploty. Při tomto rozšiřování se opět vyskytla řada problémů. Prvním problémem byla verze operačního systému PLC automatu. Ten se vyřešil aktualizací na vyšší verzi. Na základě aktualizace operačního systému vznikl další problém. A to s komunikací se serverem po ethernetu. Problém se nakonec podařilo vyřešit aktualizací PVI manažeru. Nevýhodou této aktualizace je, že při restartu serveru se spouští stará verze PVI manažeru a je třeba tento manažer ukončit a spustit PVI manažer vyšší verze. Tato nevýhoda se dá odstranit aktualizací celého komunikačního systému na vyšší verzi. Tento způsoby byl otestován v testovací aplikaci. Aktualizace na druhou stranu přináší problém nutnosti opětovné konfigurace komunikace přes OPC. A to od nastavení OPC v PLC automatu až po konfiguraci OPC na serveru a ve vizualizačních aplikacích. V neposlední řadě byla vytvořena vizualizační aplikace pro sledování změřených dat a možnost nastavování parametrů měření.

V současné době je rozšířený systém v provozu na budově VEC2 (Výzkumné energetické centrum 2).

9 Literatura

[1] KOZIOREK, Jiří, CHROMČÁK, Libor. *Logické systémy řízení: Učební text, příklady pro cvičení.* VŠB – Technická Univerzita Ostrava, 2007. ISBN 978-80-248-1490-2.

[2] *X20 User's Manual* [online]. 2007. [citováno 31.9.2010]. <http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000069247/MAX20-ENG_V200_08_2007.pdf>.

[3] *DS18B20 datasheet* [online]. 2001. revize 2008. [citováno 15.9.2010]. <http://pdfserv.maxim-ic.com/en/ds/DS18B20.pdf>.

[4] *User Manual ADA-101W* [online]. 2001. [citováno 15.9.2010]. <http://www.cel-mar.pl/files/io/io_ada-101W_en.pdf>.

[5] *1-Wire Devices* [online]. 2010. [citováno 31.9.2010]. <http://www. maxim-ic.com/products/1-wire/>.

[6] Láník, Vladimír. *MicroLAN – A jde to i s jedním vodičem!* [online]. 2005. [citováno 31.10.2010]. <http://www.hw.cz/Rozhrani/ART1240-MicroLan---A-jde-to-i-s-jednim-vodicem.html>.

[7] Malý, Martin. *Sběrnice 1-Wire* [online]. 2004. [citováno 31.10.2010]. <http://www.hw.cz/Rozhrani/ART1215-Sbernice-1-Wire™.html>.

[8] Musiol, Tomáš, Černoch, Lukáš. *Systémy pro měření a monitorování teplot hlubinných vrtů tepelných čerpadel Nové Auly VŠB-TUO Ostrava: Semestrální projekt na fakultě elektrotechniky a informatiky VŠB-TUO Ostrava.*

[9] *DS2480B datasheet* [online]. 2001. revize 2009 [citováno 31.9.2010]. <http://datasheets.maxim-ic.com/en/ds/DS2480.pdf>.

[10] *iButton product line* [online]. 2010. [citováno 27.12.2010]. <http://www.maxim-ic.com/products/ibutton/ >.

[11] *Application Note 192 – Using the DS2480B Serial 1-Wire line Driver* [online]. 2001. revize 2003 [citováno 27.2.2011]. <http://pdfserv.maxim-ic.com/en/an/AN192.pdf >.

10 Seznam příloh

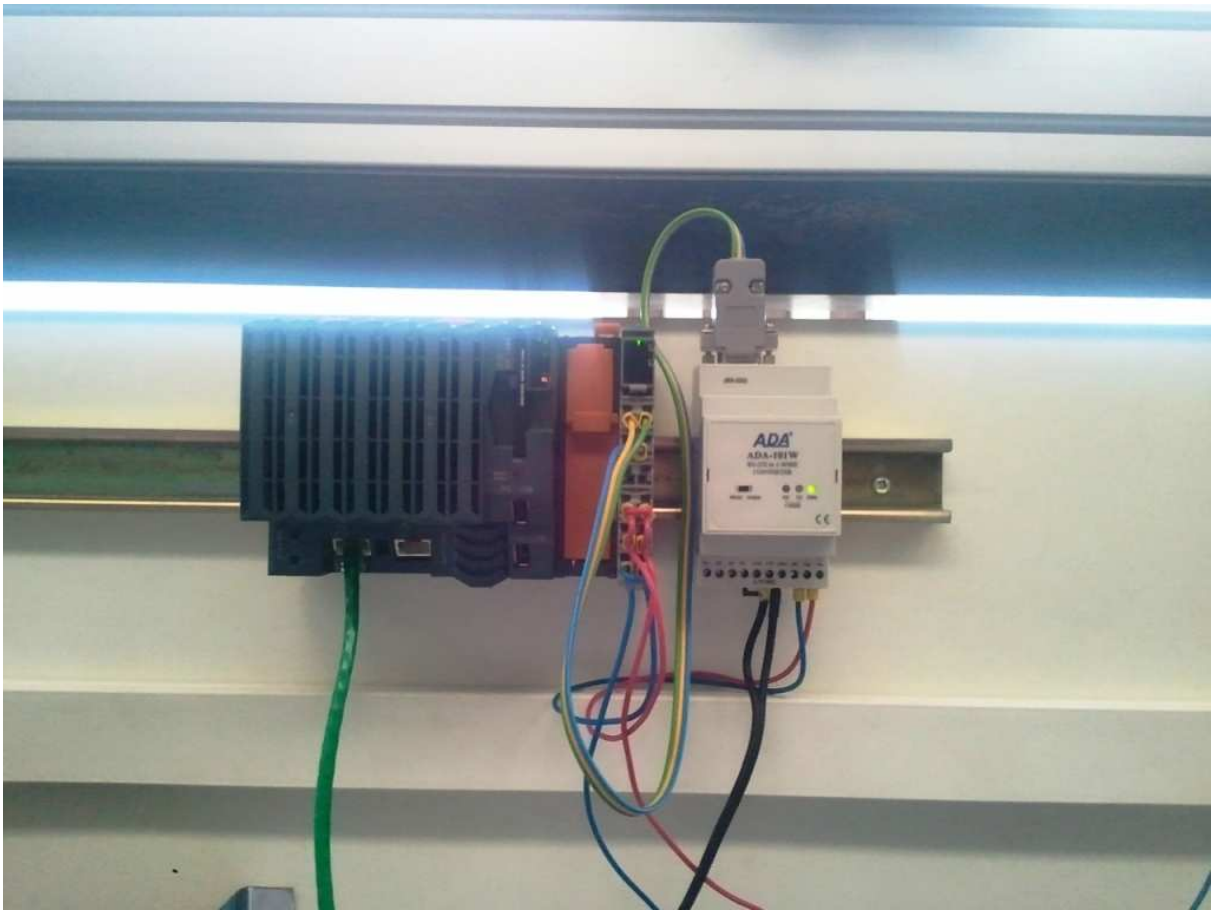
PŘÍLOHA I – Tabulka chybových konstant knihovny TempRSWire
PŘÍLOHA II – Fotka testovacího modelu
PŘÍLOHA III – Fotka reálného provozu
PŘÍLOHA IV – Nastavení OPC pro testovací aplikaci
PŘÍLOHA V – Nastavení OPC pro reálný provoz
PŘÍLOHA VI – Knihovna TempRSWire
PŘÍLOHA VII – Elektronická příloha (DVD)

PŘÍLOHA I – Tabulka chybových konstant knihovny TempRSWire

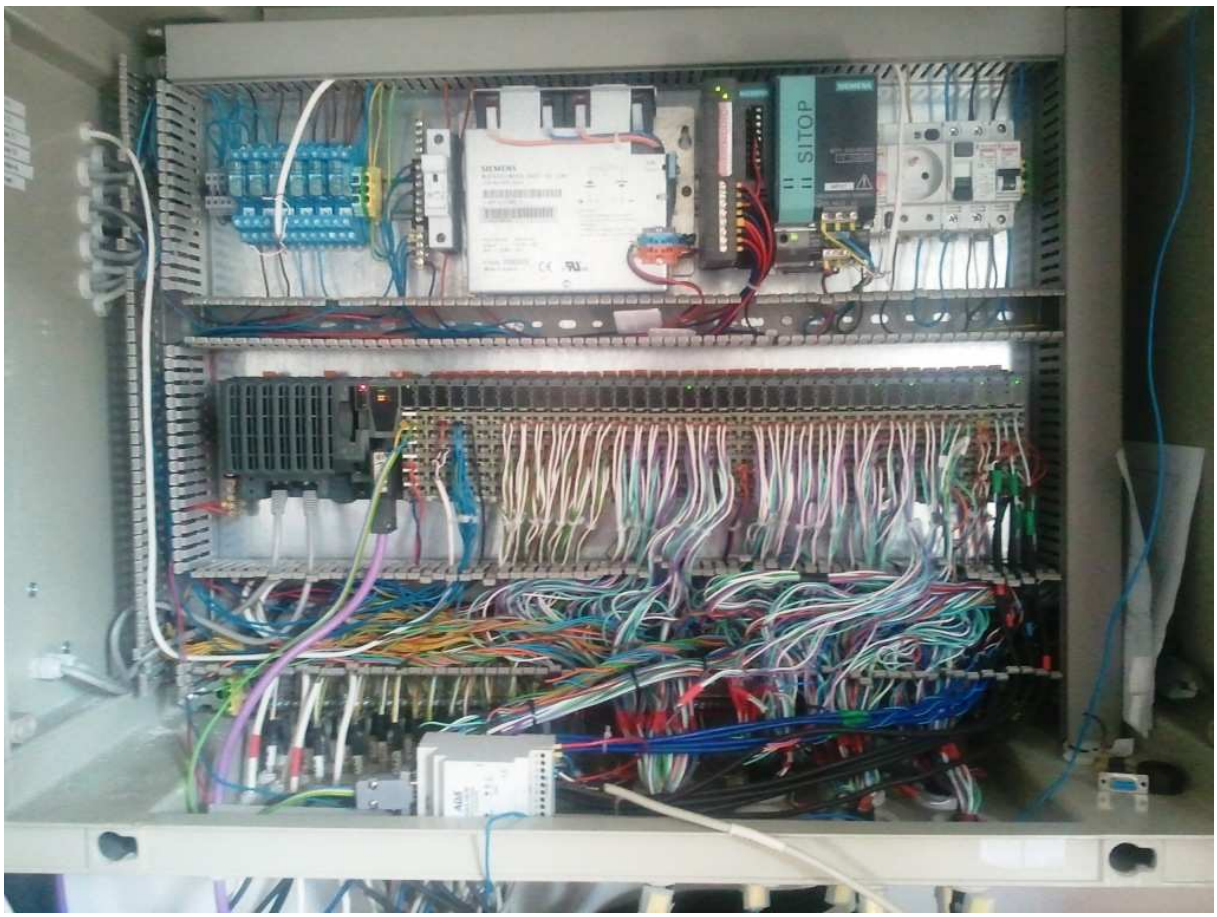
Číslo chyby	Popis chyby
0	Nenastala žádná chyba
1	Chyba struktury FRM_xopen. Chyba ve funkci OpenCOM.
2	Chyba struktury FRM_close. Chyba ve funkci CloseCOM.
3	Chyba struktury FRM_gbuf. Chyba ve funkci FlushCOM nebo WriteCOM.
4	Chyba struktury FRM_write. Chyba ve funkci WriteCOM.
5	Chyba struktury FRM_robuf. Chyba ve funkci FlushCOM nebo WriteCOM.
6	Chyba struktury FRM_read. Chyba ve funkci ReadCOM.
7	Chyba struktury FRM_rbuf. Chyba ve funkci ReadCOM.
8	Chyba modifikace nastavení sériového portu. Chyba ve funkci SetBaudCOM.
9	Chybná odpověď z převodníku. Chyba ve funkci DS2480Detect.
10	Chybná odpověď z převodníku. Chyba ve funkci OWReset.
11	Chyba funkce OWReset a chyba při synchronizaci komunikace.
12	Chybná odpověď z převodníku. Chyba ve funkci OWLevel.
13	Chybná odpověď z převodníku. Chyba ve funkci OWLevel.
14	Chyba funkce OWLevel a chyba při synchronizaci komunikace.
15	Chyba při resetu 1-Wire sítě ve funkci OWNext.
16	Chybná CRC hodnota ve funkci OWNext.
17	Chyba funkce OWNext a chyba při synchronizaci komunikace.
18	Chyba při rekonstrukci konverzního bytu ve funkci OWConvert.
19	Chyba funkce OWConvert a chyba při synchronizaci komunikace.
20	Chyba při resetu 1-Wire sítě ve funkci OWMATCHROM.
21	Chyba při rekonstrukci ROM kódu ve funkci OWMATCHROM.
22	Chybná odpověď na příkaz přeskočení ROM kódu. Chyba ve funkci OWMATCHROM.
23	Chyba funkce OWMATCHROM a chyba při synchronizaci komunikace.
24	Chyba při resetu 1-Wire sítě ve funkci OWSkipMATCHROM.
25	Chybná odpověď na příkaz přeskočení ROM kódu. Chyba ve funkci OWSkipMATCHROM.
26	Chyba funkce OWSkipMATCHROM a chyba při synchronizaci komunikace.
27	Chyba při nastavování úrovně 1-Wire sítě pomocí funkce OWWriteConfig.
28	Chyba přístupu na 1-Wire zařízení ve funkci OWWriteConfig.
29	Chybná odpověď na příkaz zápisu do scratchpad paměti. Chyba ve funkci OWWriteConfig.
30	Chyba funkce OWWriteConfig a chyba při synchronizaci komunikace.
31	Chyba při nastavování úrovně 1-Wire sítě pomocí funkce OWWriteConfigAll.
32	Chyba přístupu na 1-Wire zařízení ve funkci OWWriteConfigAll.
33	Chyba funkce OWWriteConfigAll a chyba při synchronizaci komunikace.
34	Chyba při nastavování úrovně 1-Wire sítě pomocí funkce OWReadScratchpad.
35	Chyba přístupu na 1-Wire zařízení ve funkci OWReadScratchpad.
36	Chybná odpověď na příkaz čtení scratchpad paměti. Chyba ve funkci OWReadScratchpad.
37	Chybná CRC hodnota ve funkci OWReadScratchpad.
38	Chyba funkce OWReadScratchpad a chyba při synchronizaci komunikace.
39	Chyba při nastavování úrovně 1-Wire sítě pomocí funkce OWCopyScratchpad.
40	Chyba přístupu na 1-Wire zařízení ve funkci OWCopyScratchpad.
41	Chyba při vykonávání funkce OWConvert ve funkci OWCopyScratchpad.
42	Chyba funkce OWCopyScratchpad a chyba při synchronizaci komunikace.
43	Chyba při nastavování úrovně 1-Wire sítě pomocí funkce OWCopyScratchpadAll.

44	Chyba přístupu na 1-Wire zařízení ve funkci OWCopyScratchpadAll.
45	Chyba při vykonávání funkce OWConvert ve funkci OWCopyScratchpadAll.
46	Chyba funkce OWCopyScratchpadAll a chyba při synchronizaci komunikace.
47	Chyba při resetu 1-Wire sítě ve funkci OWPresentDevice.
48	Chyba funkce OWPresentDevice a chyba při synchronizaci komunikace.
49	Chyba přístupu na 1-Wire zařízení ve funkci OWReadTemp.
50	Chyba inicializace (konverze) teploty ve funkci OWReadTemp.
51	Chyba při čtení scratchpad paměti ve funkci OWReadTemp.
52	Chyba funkce OWReadTemp a chyba při synchronizaci komunikace.

PŘÍLOHA II – Fotka testovacího modelu



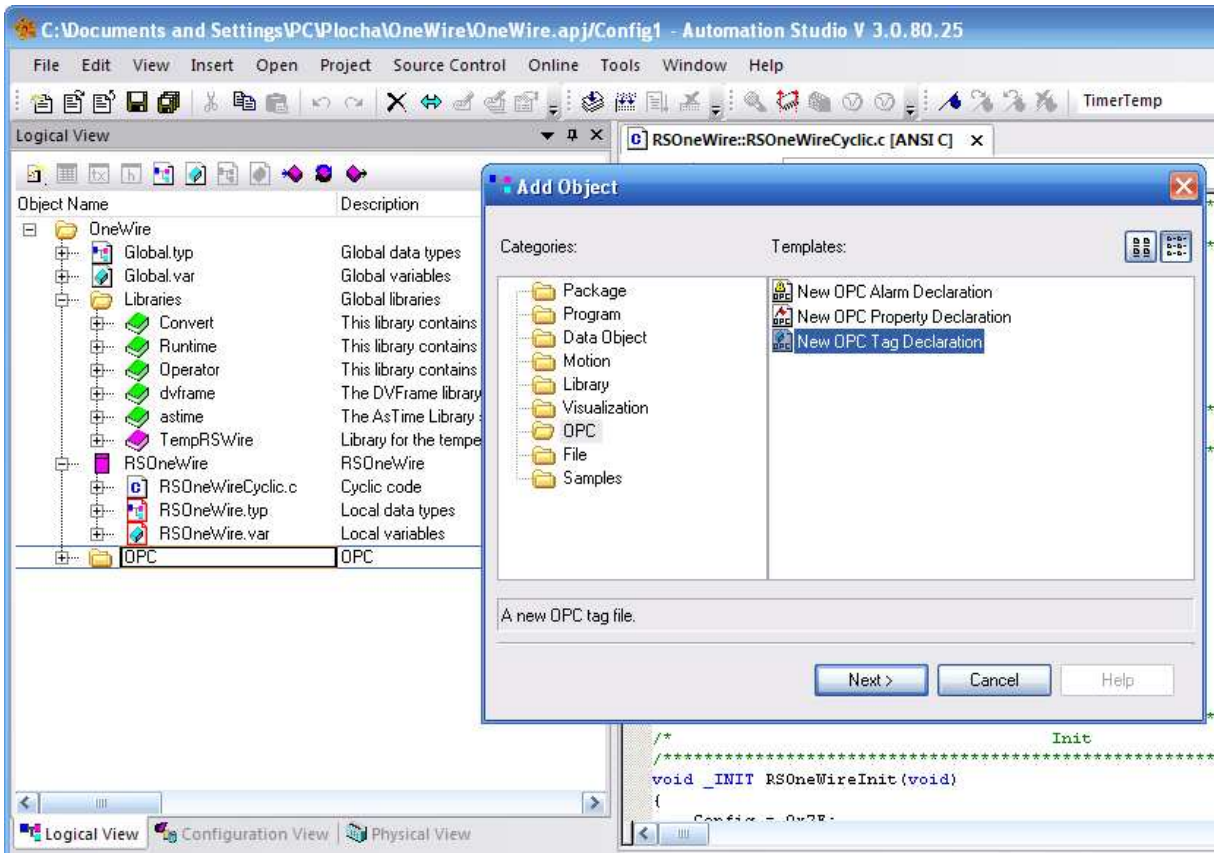
PŘÍLOHA III – Fotka reálného provozu



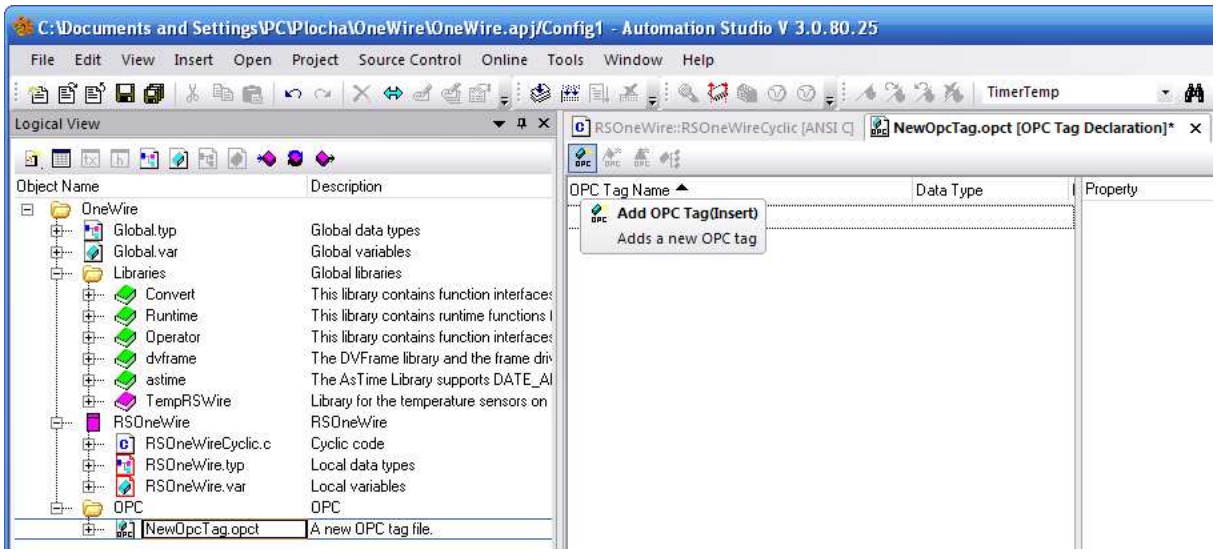
PŘÍLOHA IV – Nastavení OPC pro testovací aplikaci

Nastavení začíná v aplikaci vyvíjené v prostředí Automaton Studia:

1. Volba záložky Logical View.

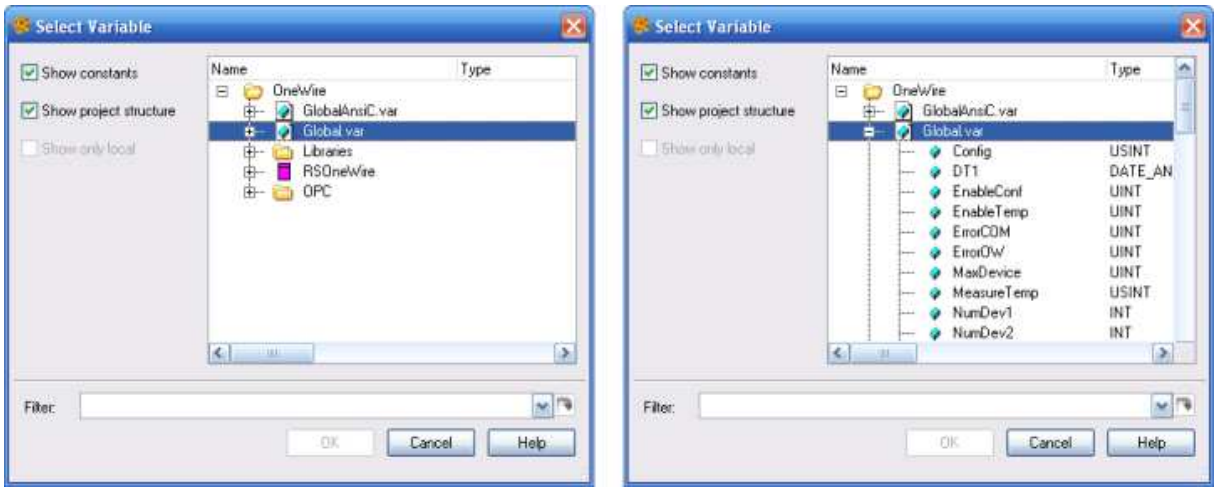


Obr.IV.1: Tvorba nového OPC tag souboru



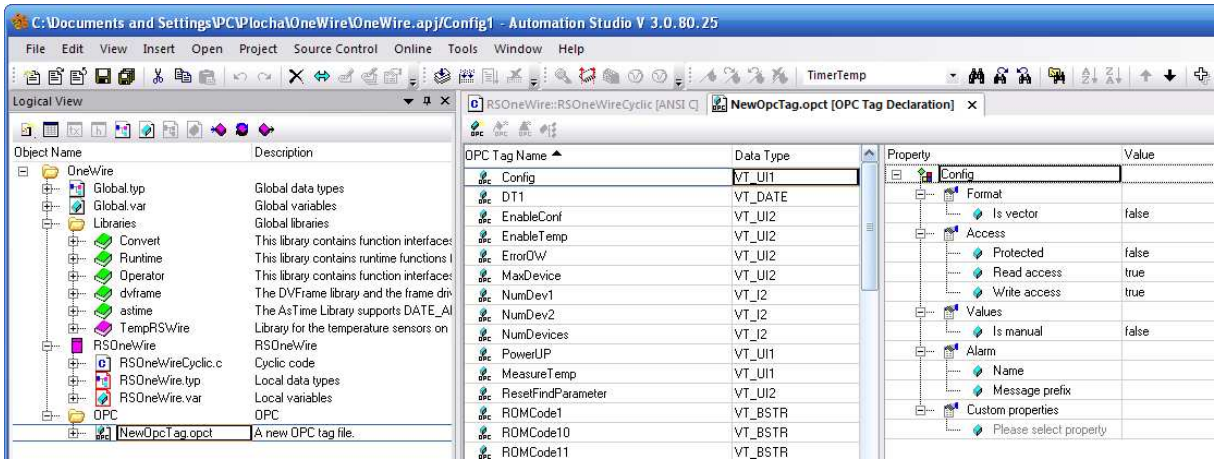
Obr.IV.2: Přidání OPC tagu do tag souboru

2. Otevření okna Add Objekt – Pravým kliknutím myši na položku záložky Logical View se vyvolá lišta voleb. V ní se volí Add Objekt. Tím se otevře okno Add Objekt.
3. Tvorba nového OPC tag souboru (viz. Obrázek IV.1) – V bloku Categories okna Add Objekt se volí záložka OPC, tím se zviditelní nabídka možných voleb. Tyto volby jsou v bloku Tamplates. Zde se zvolí New OPC Tag Declaration a vytvoří se nový OPC tag soubor (NewOpcTag.opct). Příklad vytvořeného tag souboru je na obrázku IV.2.
4. Otevření vzniklého OPC tag souboru – Dvojitým kliknutím na položku vzniklého OPC tag souboru (NewOpcTag.opct) se otevře požadované okno.
5. Přidání OPC tagu do tag souboru (viz. Obrázek IV.2) – Přidání nového tagu se realizuje stiskem položky Add OPC Tag a následným přidáním konkrétní proměnné (Pomocí položky Select Variable – viz. Obrázek IV.3).



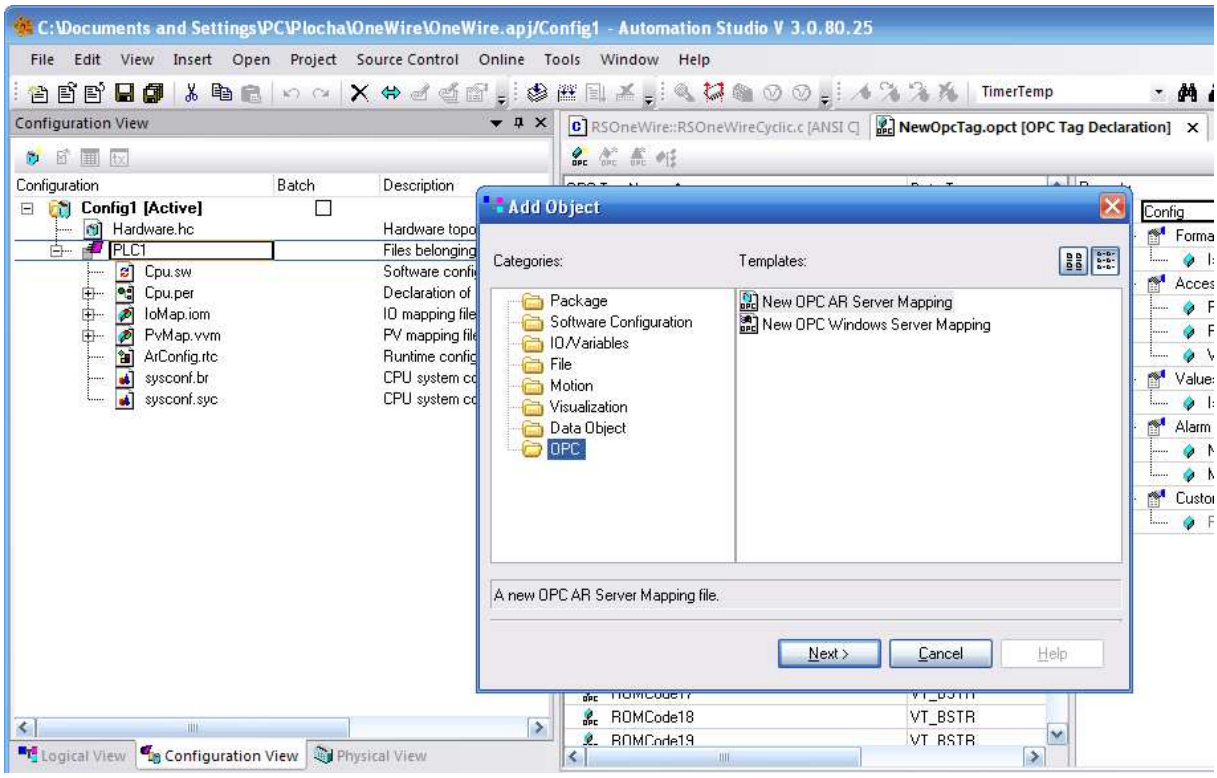
Obr.IV.3: Okno Select Variable

6. Nastavení jednotlivých OPC tagů (viz. Obrázek IV.4) – Zvolí se konkrétní OPC tag a v pravém okně tag souboru se objeví položky pro možné nastavení. Stačí si zvolit konkrétní položku. Pro začátek stačí nastavit, jestli je proměnná pro čtení nebo pro čtení a zápis (položka Write Access).

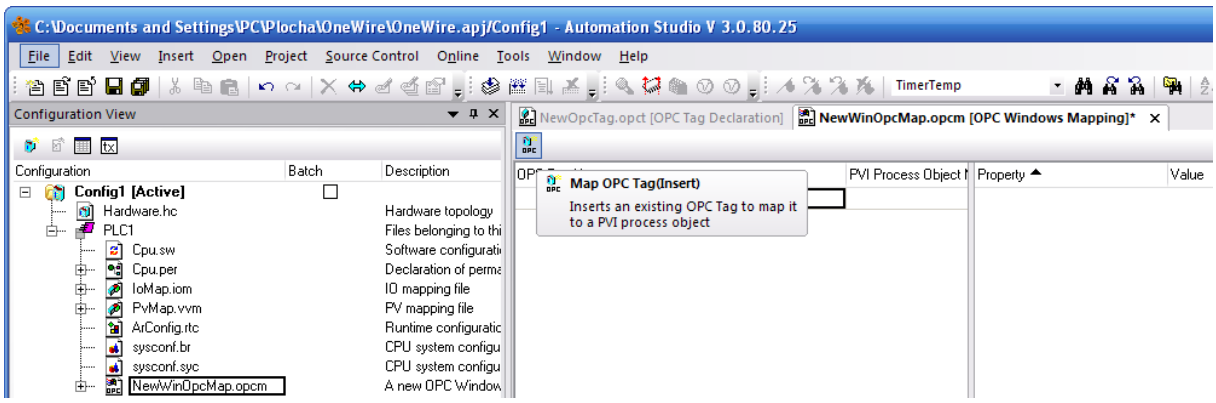


Obr.IV.4: Nastavení OPC tagů

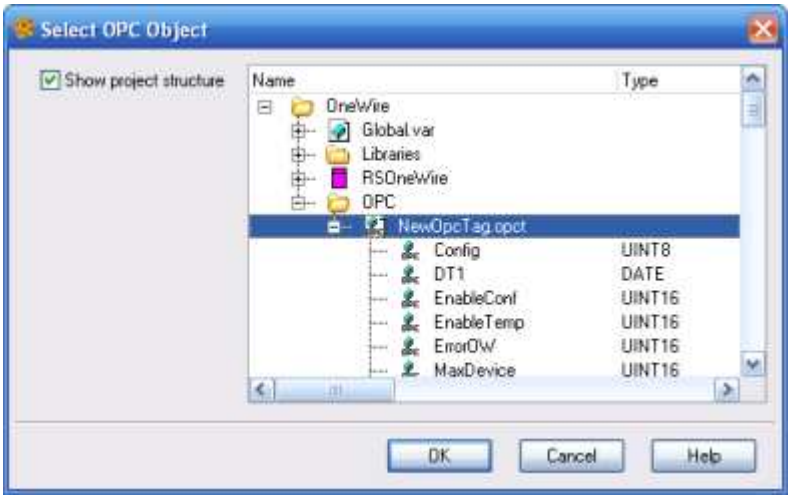
7. Volba záložky Configuration View.
8. Otevření okna Add Objekt – Okno se otevírá stejným způsobem jako u bodu 2.
9. Tvorba nové mapy OPC AR serveru (viz. Obrázek IV.5) – V bloku Categories okna Add Objekt se volí záložka OPC, tím se zviditelní nabídka možných voleb. Tyto volby jsou v bloku Tamplates. Zde se zvolí New OPC AR Server Mapping a vytvoří se nová OPC mapa (NewWinOpcMap.opcm).
10. Otevření vzniklého mapovacího OPC souboru – Dvojitým kliknutím na položku vzniklého mapovacího souboru (NewWinOpcMap.opcm) se otevře požadované okno.
11. Přidání Map OPC tagu do mapovacího souboru (viz. Obrázek IV.6) – Přidání nového tagu se realizuje stiskem položky Map OPC Tag a následným přidáním konkrétního OPC objektu (Pomocí položky Select OPC Object – viz. Obrázek IV.7).



Obr.IV.5: Tvorba nové mapy OPC AR Serveru

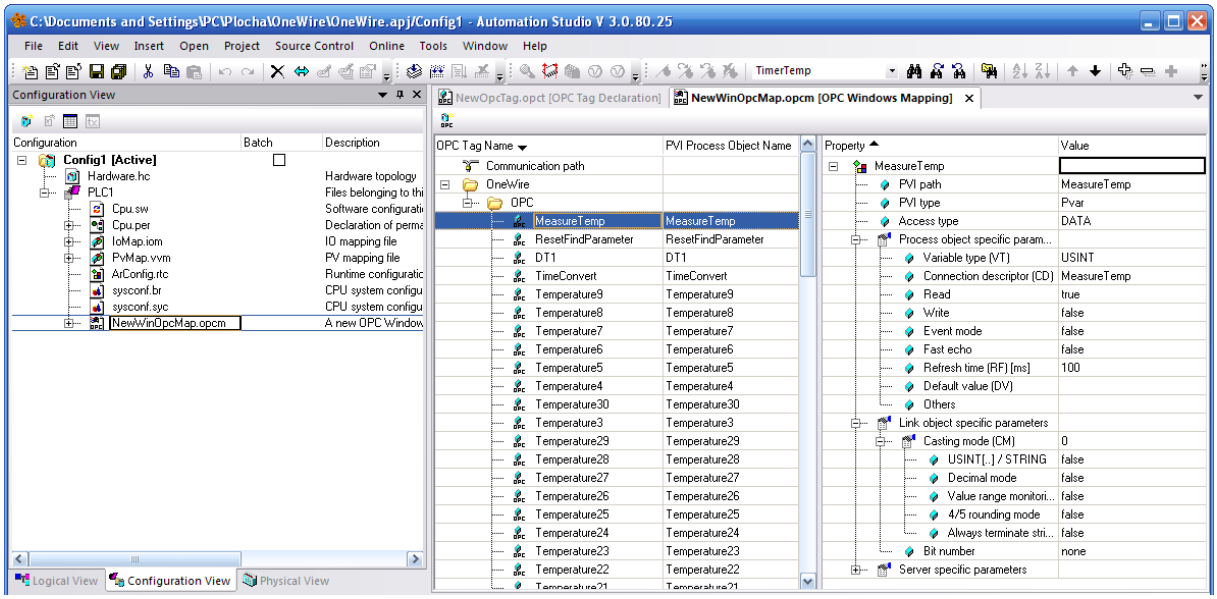


Obr.IV.6: Přidání Map OPC tagu do souboru mapy OPC AR serveru



Obr.IV.7: Přidání Map OPC tagu do souboru mapy OPC AR serveru

12. Nastavení jednotlivých Map OPC tagů (viz. Obrázek IV.8) – Zvolí se konkrétní Map OPC tag a v pravém okně mapovacího souboru se objeví položky pro možné nastavení. Stačí si zvolit konkrétní položku. Pro začátek stačí nastavit obnovovací čas (položka Refresh Time).

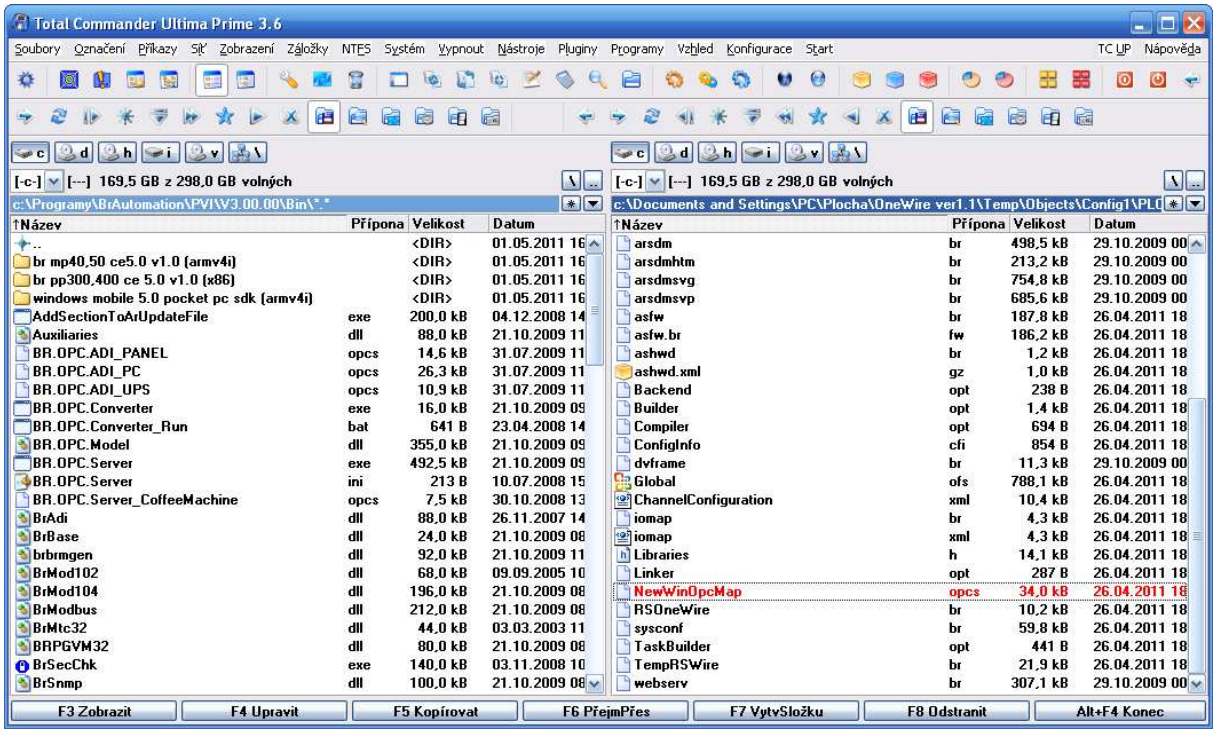


Obr.IV.8: Nastavení Map OPC tagů

13. Nastavení komunikace – Pro nastavení komunikace je nutné zvolit položku Communication patch (viz. Obrázek IV.8). Volbou této položky se v pravém okně mapovacího souboru objeví položky pro možné nastavení. V položce CPU je nutné nastavit IP adresu (položka Destination IP address/Hostname – 158.196.133.99), adresu destinace (položka Destination address - 7) a číslo portu (položka Port number – 11159). Číslo portu musí být stejné jako lokální číslo portu (položka Local port number – 11159) položky Tcpip. Čísla se mohou lišit v případě použití simulátoru.

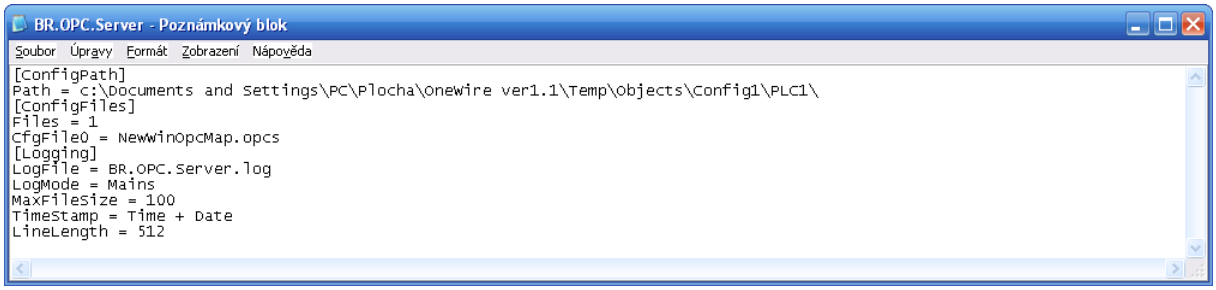
Nastavení komunikace ve 13 bodu končí nastavování OPC z hlediska Automation Studia. Další nastavování je z hlediska PVI:

1. Zkopírování mapovacího souboru OPC serveru (NewWinOpcMap.opcm) do složky bin aplikace PVI (viz. Obrázek IV.9).



Obr.IV.9: Kopírování mapovacího souboru OPC serveru

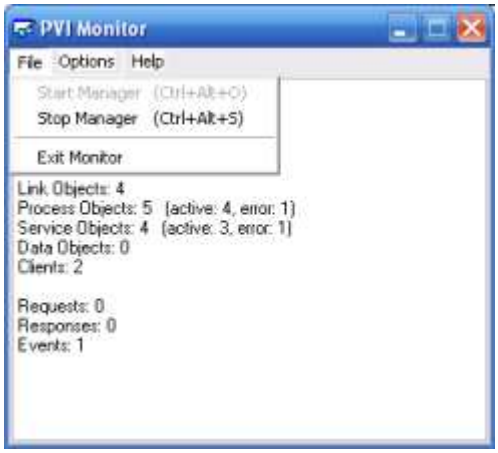
2. Konfigurace inicializačního souboru (BR.OPC.Server) pro nastavení OPC serveru PVI aplikace – V souboru je třeba změnit cestu (položka Path) k mapovacímu souboru (NewWinOpcMap.opcm) a název u položky CfgFile0 je potřeba napsat celý název mapovacího souboru (viz. Obrázek IV.10). Změnou tohoto nastavení PVI aplikace konfiguruje OPC server podle mapovacího souboru generovaného v prostředí Automation Studia.



Obr.IV.10: Soubor BR.OPC.Server

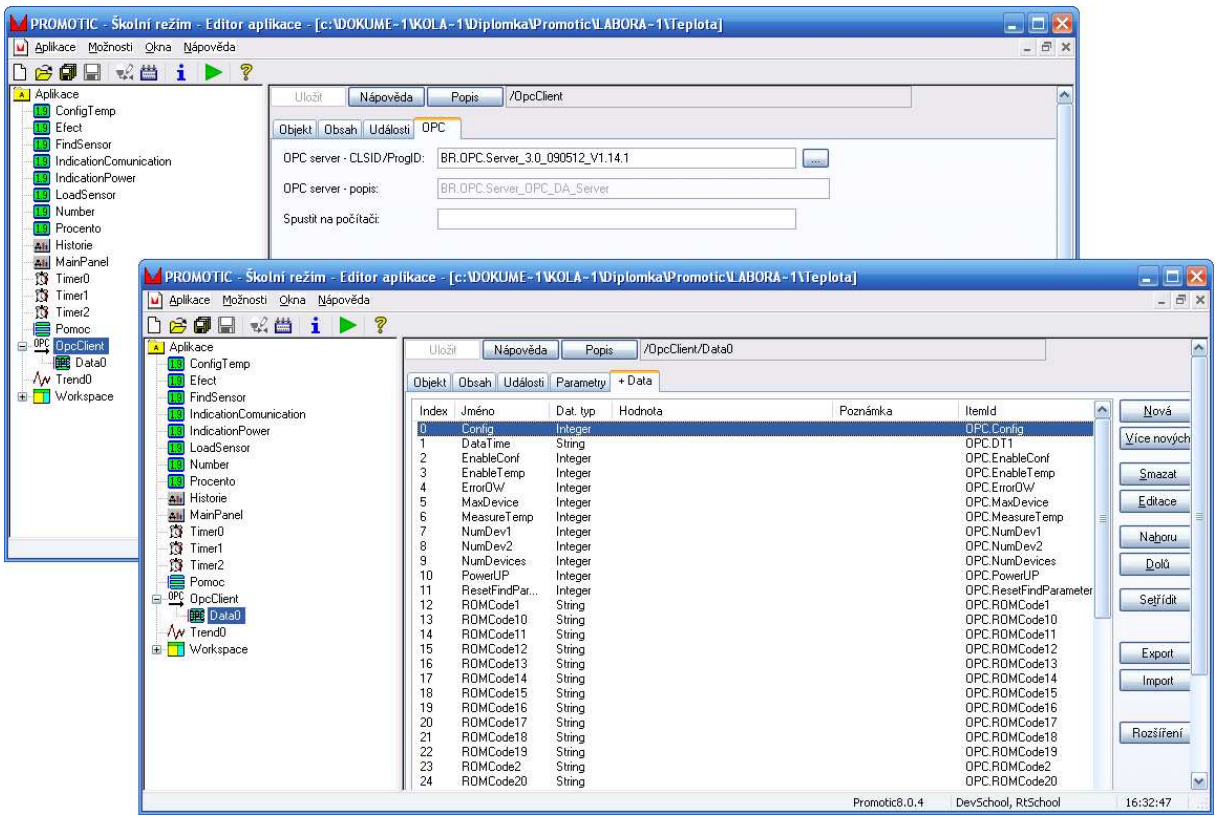
3. Restart PVI manažeru – Restart se provádí kvůli inicializaci nastavení OPC serveru. Pro tento účel stačí otevřít PVI Monitor (viz. Obrázek IV.11) a v menu File zvolit položku

Stop Manager (komunikace se ukončí). Následně se musí zvolit položka Start Manager pro opětovnou aktivaci komunikace.



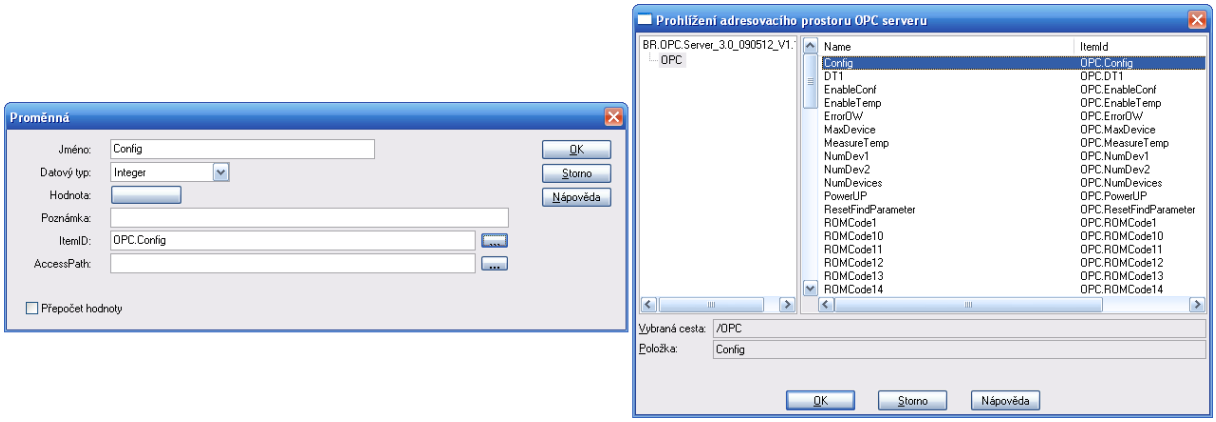
Obr.IV.11: PVI Monitor

3 bodem končí nastavování PVI aplikace. Další nastavování je již v aplikaci vizualizace. Vizualizace se tvoří v prostředí Promoticu. Pro nastavení OPC v Promoticu se používá blok OPC Client a blok OPC Data (viz. Obrázek IV.12). OPC Client (OpcClient) obsahuje pouze nastavení požadovaného OPC serveru (záložka OPC a položka OPC server – CLSID/ProgID). V OPC Data (Data0) se vytvářejí konkrétní proměnné obsahující OPC data z vybraného serveru. K tomuto účelu slouží záložka Data.



Obr.IV.12: Nastavení OPC v Promoticu

Záložka Data bloku OPC Data obsahuje tlačítka pro vyvolání konkrétní operace (editace, nová apod.). Po aktivaci jedné z nich se otevře nabídka Proměnná (viz. Obrázek IV.13). Pomocí této nabídky se vytvoří nebo upraví požadovaná proměnná. Důležitá je položka NewID. Po její aktivaci se otevře nabídka, pomocí níž se vybírá konkrétní proměnná poskytující OPC server (viz. Obrázek IV.13).

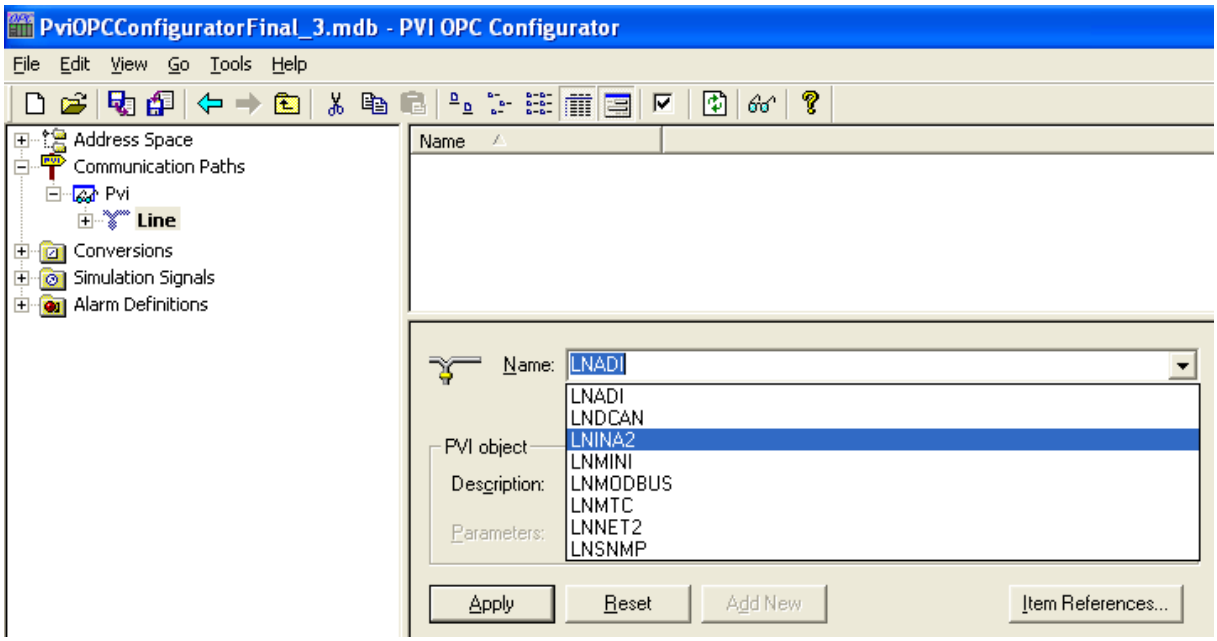


Obr.IV.13: Nastavení OPC proměnné

PŘÍLOHA V – Nastavení OPC pro reálný provoz

Pro nastavení OPC v reálném provozu se používá aplikace PVI. Vlastní OPC parametry (OPC proměnné, OPC server apod.) se nastavují v PVI OPC konfigurátoru. PVI OPC konfigurátor je součástí PVI Dveloper kitu. Nastavení komunikační cesty PVI OPC konfigurátoru:

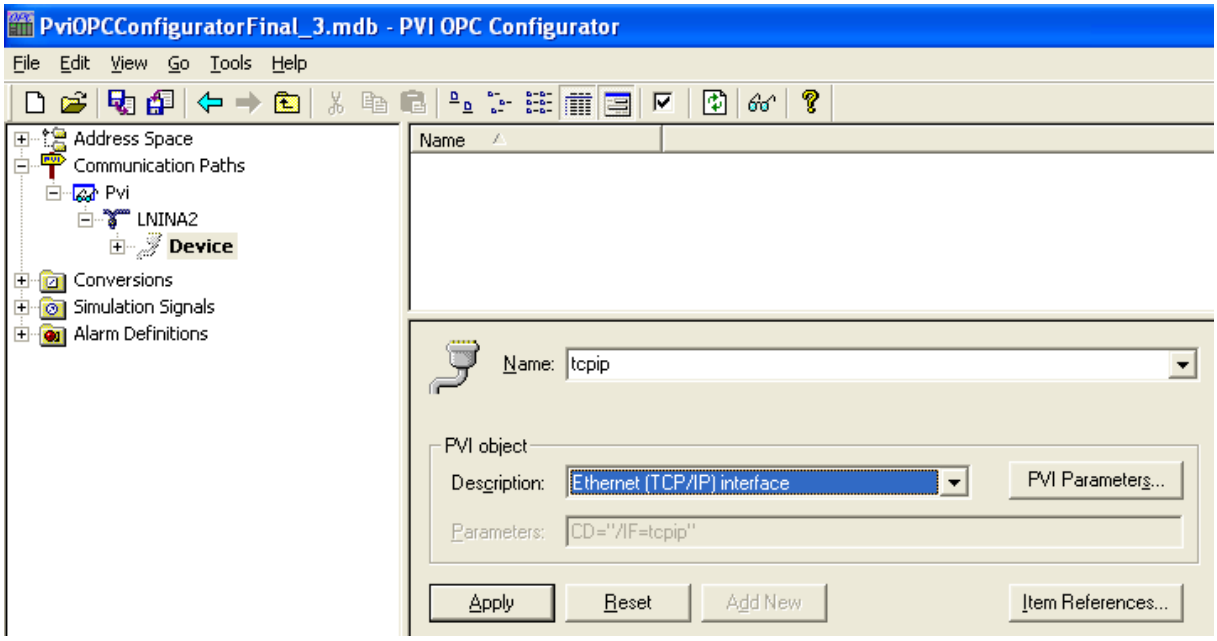
1. Vytvoření PVI připojení – Kliknutím pravého tlačítka myši na záložku Communication Paths se otevře nabídka a v ní se zvolí New/PVI Connection. V pravém okně se otevře nabídka nastavení. V této nabídce stačí změnit čas komunikace (položka Communication timeout) na 1 sekundu. Popřípadě se může změnit IP adresa. V případě, že je vytvořena jen jedna komunikační cesta stačí ponechat automaticky generovanou IP adresu. V případě více komunikačních cest je nutné IP adresu změnit. Ta se mění kliknutím levým tlačítkem myši na položku Connect to remote PVI.
2. Vytvoření nové linky (viz. Obrázek V.1) – Kliknutím pravého tlačítka myši na položku PVI připojení se otevře nabídka a v ní se zvolí New/Line. V pravém okně se otevře nabídka nastavení. V položce Name okna nastavení se musí zvolit název linky. Levým tlačítkem myši se klikne na šipku, tím se otevře nabídka s názvy linek. Zde se volí položka LNINA2.



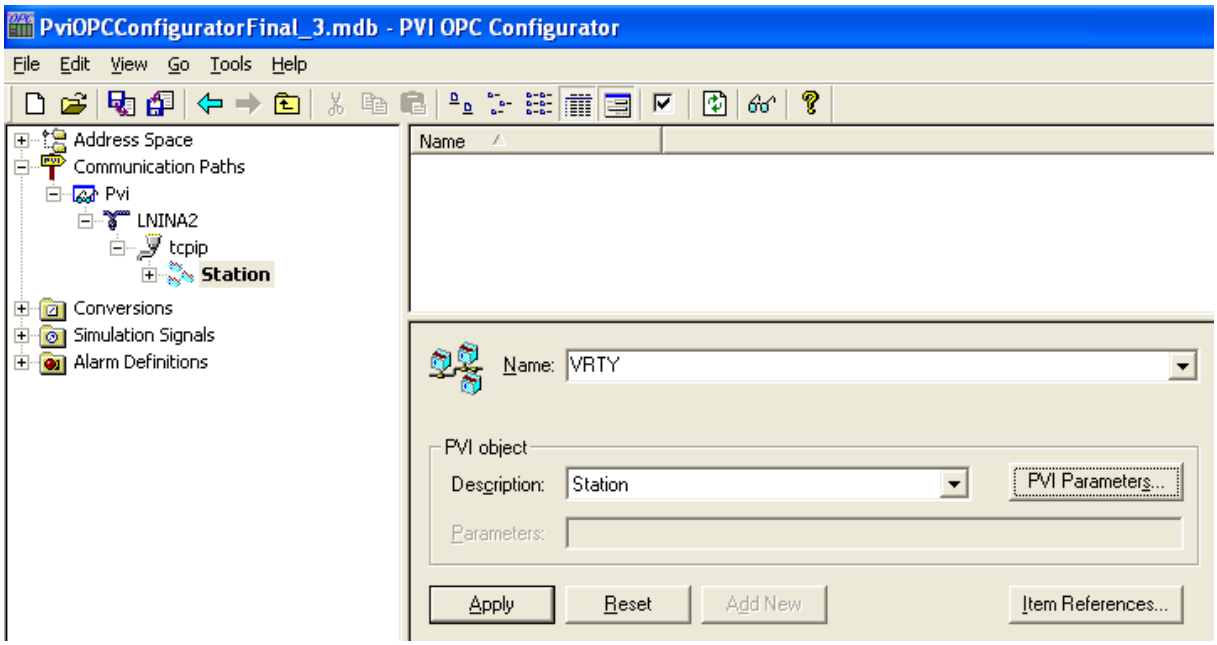
Obr.V.1: Nastavení nové linky

3. Vytvoření nového zařízení (viz. Obrázek V.2) – Kliknutím pravého tlačítka myši na položku nové linky se otevře nabídka a v ní se zvolí New/Device. V pravém okně se otevře nabídka nastavení. V položce PVI object/Description okna nastavení se musí zvolit typ zařízení. Levým tlačítkem myši se klikne na šipku a tím se otevře nabídka s typy zařízení. Zde se volí položka Ethernet (TCP/IP) interface.
4. Vytvoření nové položky station (viz. Obrázek V.3) – Kliknutím pravého tlačítka myši na položku nového zařízení se otevře nabídka a v ní se zvolí New/Station. V pravém okně se

otevře nabídka nastavení. V položce Name okna nastavení se musí vytvořit název (stačí napsat název VRTY).



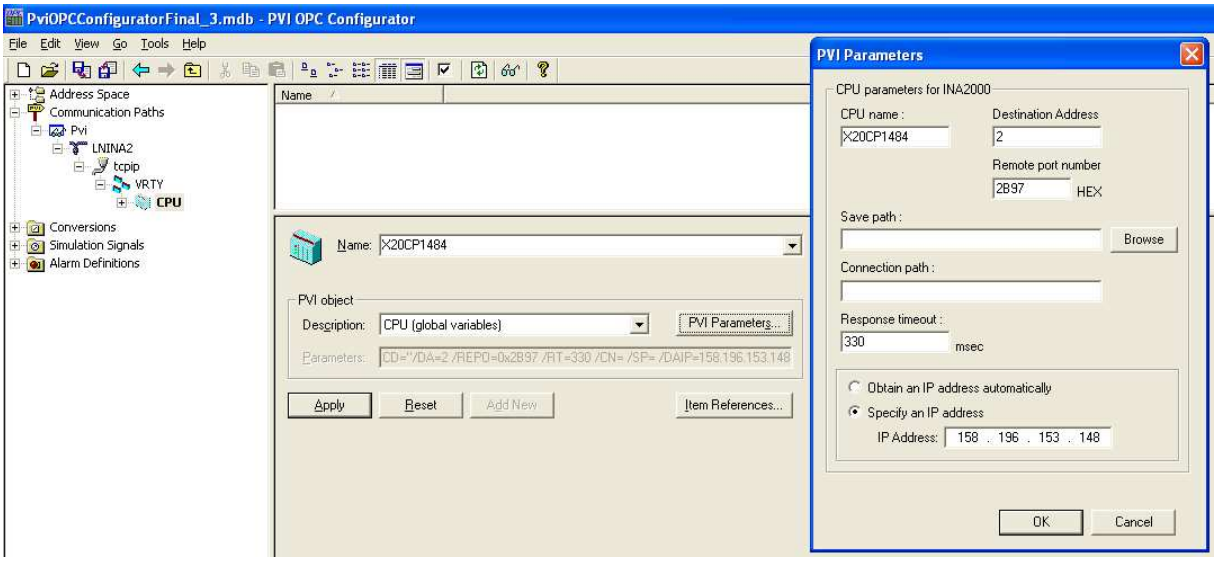
Obr.V.2: Nastavení nového zařízení



Obr.V.3: Nastavení nové položky station

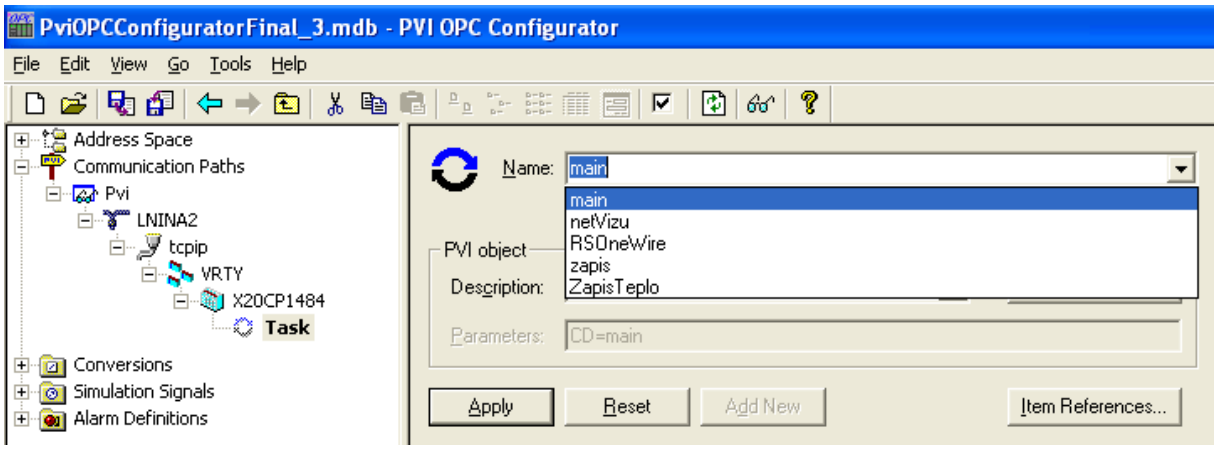
5. Vytvoření nové CPU (viz. Obrázek V.4) – Kliknutím pravého tlačítka myši na položku nového station (VRTY) se otevře nabídka a v ní se zvolí New/CPU. V pravém okně se otevře nabídka nastavení. Pro nastavení CPU stačí kliknout na tlačítko PVI Parameters. Otevře se okno PVI Parameters. V okně PVI Parameters je nutné nastavit název CPU (položka CPU name – X20CP1784), IP adresu (položka IP address – 158.196.153.148) a

adresu destinace (položka Destination address - 2). Aby bylo možné nastavit IP adresu je nutné aktivovat položku Specifi an IP address (viz. Obrázek V.4).



Obr.V.4: Nastavení nové CPU

6. Přidání tasku (viz. Obrázek V.5) – Kliknutím pravého tlačítka myši na položku nového CPU se otevře nabídka a v ní se zvolí New/Task. V pravém okně se otevře nabídka nastavení. Levým tlačítkem myši se klikne na šipku (u položky Name), tím se otevře nabídka s možnými tasky. Zde se volí položka tasku main (V případě nutnosti se může přidat další task).



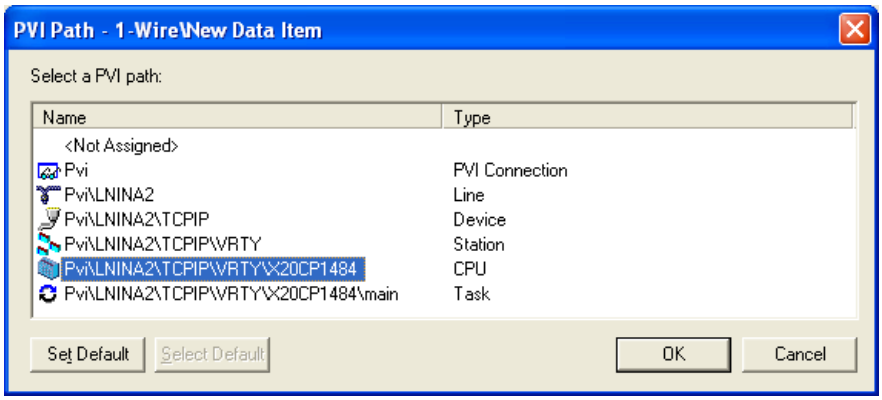
Obr.V.5: Nastavení nového tasku

6 položkou končí nastavování komunikační cesty. Nastavení proměnných OPC.PVI serveru pomocí PVI OPC konfigurátoru:

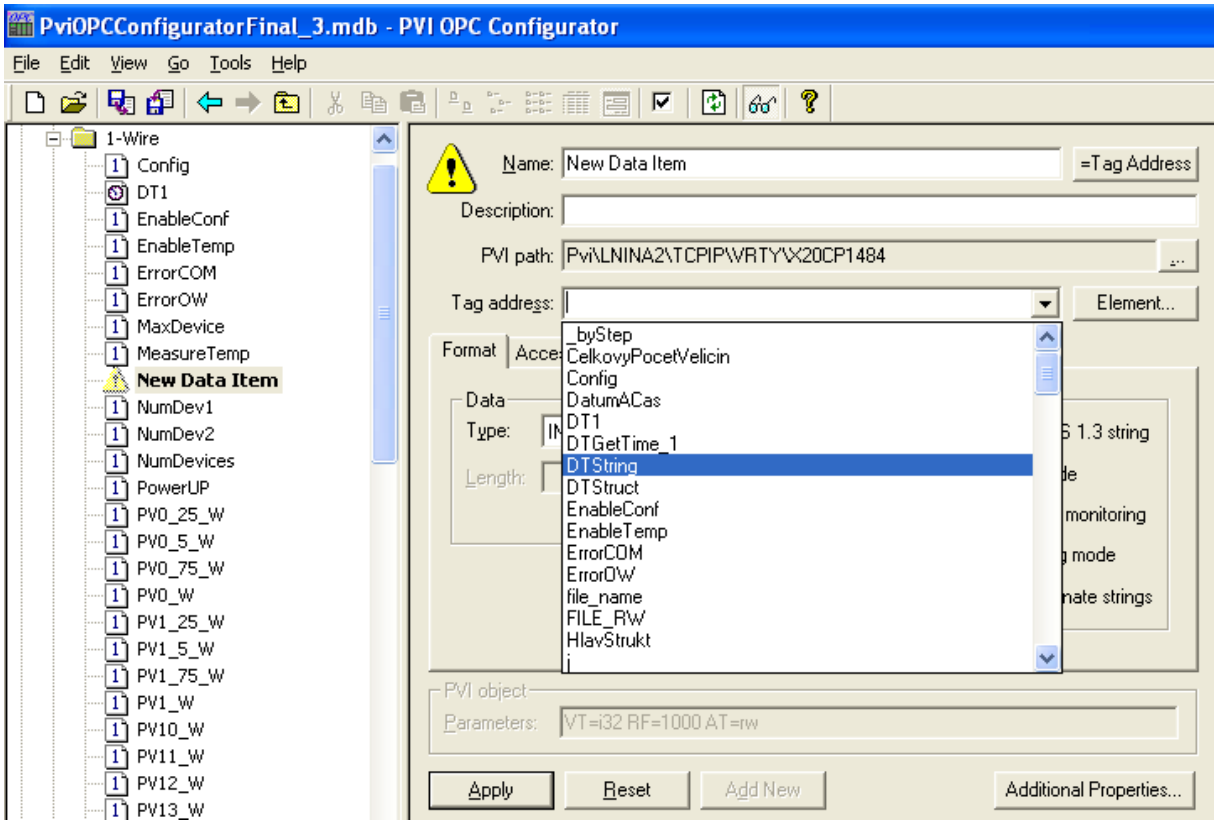
1. Vytvoření složky pro proměnné OPC.PVI serveru – Kliknutím pravého tlačítka myši na záložku Address Space se otevře nabídka a v ní se zvolí New/Folder. V pravém okně se otevře nabídka nastavení. Toto nastavení obsahuje pouze položku Name. Do této položky

se napíše požadovaný název vytvářené složky. Tímto způsobem se vytvoří složka 1-Wire a složka promene.

2. Vytvoření nové proměnné OPC.PVI serveru – Kliknutím pravého tlačítka myši na vytvořenou složku se otevře nabídka a v ní se zvolí New/Data Item. V pravém okně se otevře nabídka nastavení nové OPC proměnné.
3. Nastavení cesty k požadovaným proměnným PLC automatu – Cestu k požadovaným proměnný PLC automatu obsahuje položka PVI path. Kliknutím levého tlačítka myši na tlačítko se třemi tečkami (v oblasti PVI path) se otevře okno s možnými komunikačními cestami (viz. Obrázek V.6). Zde se vybere požadovaná cesta (podle toho jakou proměnou chceme přidat do PVI.OPC serveru).

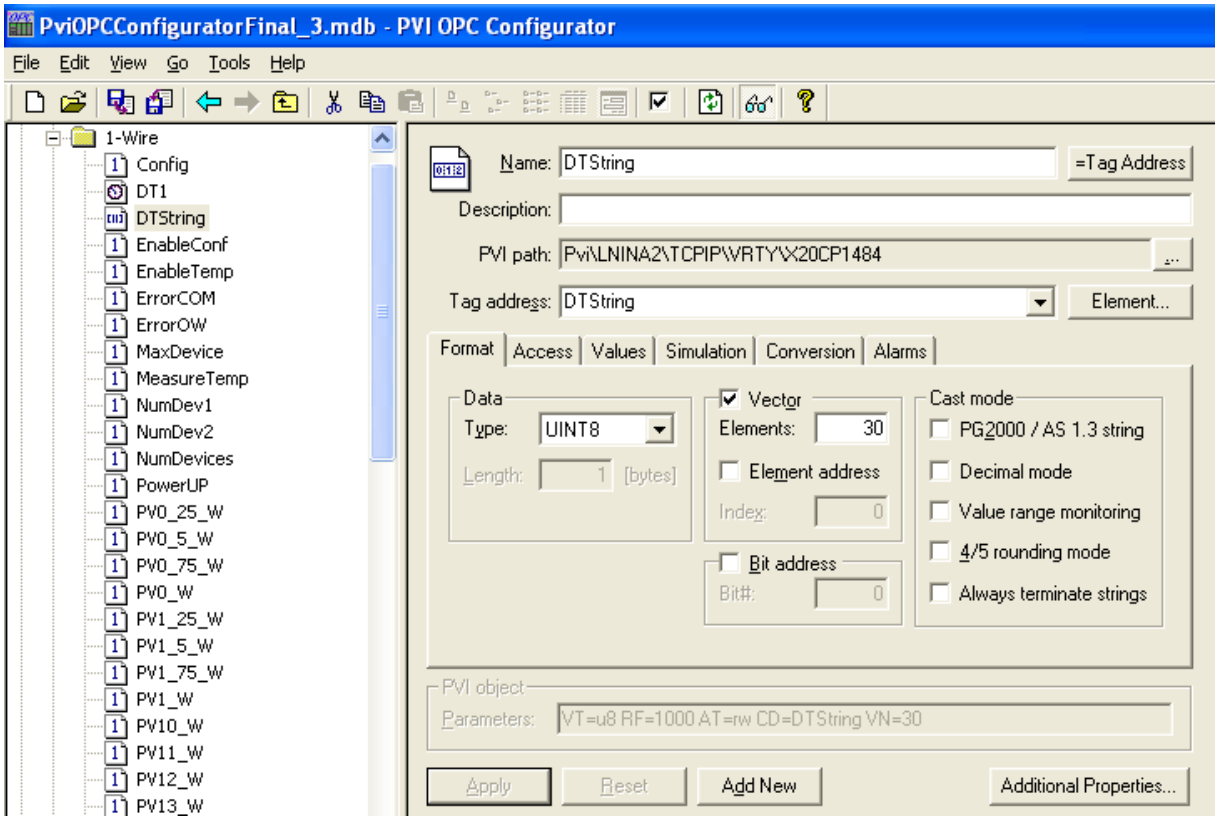


Obr.V.6: Nastavení cesty k požadované proměnné PLC automatu



Obr.V.7: Volba požadované proměnné PLC automatu

4. Výběr požadované proměnné PLC automatu (viz. Obrázek V.7) – Požadovaná proměnná se volí na základě nastavené komunikační cesty (viz. Bod 3). Proměnné PLC automatu obsahuje položka Tag address. Kliknutím levého tlačítka myši na šipku se otevře nabídka, která obsahuje dostupné proměnné PLC automatu. Požadovaná proměnná se vybere kliknutím levým tlačítkem myši na proměnnou nabídky proměnných.
5. Nastavení názvu OPC proměnné (viz. Obrázek V.8) – Název OPC proměnné obsahuje položka Name. Do této položky stáčí vepsat požadovaný název. Jednotlivé názvy vytvořených OPC proměnných jsou stejné jako názvy zdrojových proměnných PLC automatu.



Obr.V.8: Nastavení názvu OPC proměnné

5 bodem nastavování PVI aplikace končí. Další nastavení se provádí ve vizualizační aplikaci Promoticu. Postup nastavování OPC je identický jako u testovací aplikace. Rozdíl nastává jen v použitém OPC serveru.

PŘÍLOHA VI – Knihovna TempRSWire

Základní funkce knihovny TempRSWire jsou postaveny na strukturách knihovny DVFrame. Proto knihovna TempRSWire vyžaduje přítomnost knihovny DVFrame. Za tímto účelem knihovna obsahuje strukturu „RS232“, která zpřístupňuje struktury knihovny DVFrame.

Struktura „RS232“

Proměnná	Datový typ	Popis
XOpenConfigStruct	XOPENCONFIG	Definice struktury XOPENCONFIG knihovny DVFrame.
FrameXOpenStruct	FRM_xopen	Definice struktury FRM_xopen knihovny DVFrame.
FrameCloseStruct	FRM_close	Definice struktury FRM_close knihovny DVFrame.
FrameModeStruct	FRM_mode	Definice struktury FRM_mode knihovny DVFrame.
FrameGetBufferStruct	FRM_gbuf	Definice struktury FRM_gbuf knihovny DVFrame.
FrameReleaseBufferStruct	FRM_rbuf	Definice struktury FRM_rbuf knihovny DVFrame.
FrameReleaseOutputBufferStruct	FRM_robuf	Definice struktury FRM_robuf knihovny DVFrame.
FrameWriteStruct	FRM_write	Definice struktury FRM_write knihovny DVFrame.
FrameReadStruct	FRM_read	Definice struktury FRM_read knihovny DVFrame.
Identification	UDINT	Identifikační číslo portu IF.
WriteData	USINT[256]	Buffer pro zápis dat na port IF.
ReadData	USINT[256]	Buffer pro čtení dat z port IF.
SerialNum	USINT[8]	Buffer sériových čísel (ROM kódu).
StringDevice	USINT[30]	Proměnná pro definici portu IF.
StringMode	USINT[30]	Proměnná pro nastavení módu činnosti.
StringNewMode	USINT[30]	Proměnná snovým nastavením módu činnosti.
NumDev1	INT	Proměnná s počtem nalezených zařízení na 1-Wire síti.
NumDev2	INT	Proměnná pro procházení nalezených zařízení na 1-Wire síti.
ErrorCOM	UINT	Indikace chyby ve struktuře knihovny DVFrame.
ErrorOW	UINT	Indikace chyby při komunikaci po 1-Wire síti
ReadPacket	USINT[100]	Buffer přijímaných dat z 1-Wire sítě.

Tab.VI.1: Struktura RS232

Struktura se deklaruje stejným způsobem jako klasická proměnná. V případě jazyka ANSI C se před deklaraci musí umístit příkaz _GLOBAL (globální proměnná) nebo _LOCAL (lokální proměnná).

Příklad syntaxe: `_GLOBAL RS232 RSPort;` nebo `_LOCAL RS232 RSPort;`

Funkce OpenCOM

Funkce pro otevření (inicializaci) sériového portu PLC automatu. Návratová hodnota funkce je 0 nebo 1. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	PortDevice	USINT	Argument pro nastavení označení IF portu.

Tab.VI.2: Funkce OpenCOM

Syntaxe: INT OpenCOM(Struct RS232* Interface, USINT* PortDevice);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Error;

Error = OpenCOM(&RSPort, " IF1");
```

Funkce CloseCOM

Funkce pro uzavírání sériového portu PLC automatu. Návratová hodnota funkce je 0 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 2 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.3: Funkce CloseCOM

Syntaxe: INT CloseCOM(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Error;

Error = CloseCOM(&RSPort);
```

Funkce WriteCOM

Funkce pro zápis n bytů dat na sériovou linku. Návratová hodnota je 0, 3, 4 nebo 5. Jedná se o chybový příznak funkce (0 => bez chyby; 3, 4 a 5 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	WriteBuffLen	INT	Argument velikost pole dat pro odeslání na IF port.
IN	WriteBuff	USINT	Argument pole dat pro odeslání na IF port.

Tab.VI.4: Funkce WriteCOM

Syntaxe: INT WriteCOM(INT WriteBuffLen, Struct RS232* Interface, USINT* WriteBuff);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Error;

OpenCOM(&RSPort, "IF1");
Error = WriteCOM(7, &RSPort, " Hello??");
```

Funkce FlushCOM

Funkce pro vyčištění všech vyrovnávacích bufferů potřebných pro sériovou komunikaci. Návratová hodnota je 0, 3 nebo 5. Jedná se o chybový příznak funkce (0 => bez chyby; 3 a 5 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.5: Funkce FlushCOM

Syntaxe: INT FlushCOM(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Error;

OpenCOM(&RSPort, "IF1");
Error = FlushCOM(&RSPort);
```

Funkce ReadCOM

Funkce pro čtení n bytů ze sériové linky. Návratová hodnota je 0, 6 nebo 7. Jedná se o chybový příznak funkce (0 => bez chyby; 6 a 7 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	BuffLen	INT	Argument velikost pole dat pro příjem z IF portu.
OUT	ReadBuff	USINT	Argument pole dat pro příjem z IF portu.

Tab.VI.6: Funkce ReadCOM

Syntaxe: INT ReadCOM(INT BuffLen, Struct RS232* Interface, USINT* ReadBuff);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;
```

```
_GLOBAL RS232 RSPort;
_GLOBAL INT Error;
_GLOBAL USINT ReadBuffer[10];

OpenCOM(&RSPort, "IF1");
Error = ReadCOM(10, &RSPort, ReadBuffer);
```

Funkce SetBaudCOM

Funkce pro změnu nastavení přenosové rychlosti. Návratová hodnota je 0 nebo 8. Jedná se o chybový příznak funkce (0 => bez chyby; 8 => chyba).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	NewBaud	USINT	Argument pro novou přenosovou rychlost.

Tab.VI.7: Funkce SetBaudCOM

Argument NewBaud může nabývat pouze 4 hodnot (9600 kbps => 0, 19200 kbps => 2, 57600 kbps => 4 a 115200 kbps => 6).

Syntaxe: INT SetBaudCOM(USINT NewBaud, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Error;

OpenCOM(&RSPort, "IF1");
Error = SetBaudCOM(0, &RSPort);
```

Funkce DS2480Detect

Funkce pro detekci převodníku DS2480. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.8: Funkce DS2480Detect

Syntaxe: INT DS2480Detect(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OpenCOM(&RSPort, "IF1") == 0){
```

```
            Test = 1;
        }
    }
    if(Test == 1){
        Error = DS2480Detect(&RSPort);
        if(Error == 2){
            Test = 1;
        } else{
            Test = 2;
        }
    }
```

Funkce OWDetect

Funkce pro detekci převodníku DS2480 a otevření sériového portu PLC automatu. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	PortDevice	USINT	Argument pro nastavení označení IF portu.

Tab.VI.9: Funkce OWDetect

Syntaxe: INT OWDetect(USINT* PortDevice, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Test;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) != 2){
        Test = 1;
    }
}
```

Funkce OWFindDevices

Funkce pro vyhledávání zařízení na 1-Wire síti. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	FamilyCode	INT	Argument pro family kód 1-Wire zařízení. (28h => DS18B20, 10h => DS1920 a DS1820)
IN	MaxDevices	INT	Argument pro požadavek nalezení maximálního počtu 1-Wire zařízení.
OUT	ROMCode	USINT[8]	Argument pole pro ROM kód nalezeného 1-Wire zařízení.
OUT	FindDevice	INT	Argument pro počet nalezených 1-Wire zařízení.

Tab.VI.10: Funkce OWFindDevices

Syntaxe: INT OWFindDevices(INT FamilyCode, INT MaxDevices, USINT ROMCode[8], INT* FindDevice, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
	if(OWDetect("IF1", &RSPort) == 0){
		Test = 1;
	}
}
if(Test == 1){
	Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
	if(Error == 2){
		Test = 1;
	} else{
		Test = 2;
	}
}
```

Funkce OWFamilySearch

Funkce pro vyhledávání zařízení na 1-Wire síti. Návratová hodnota je 0. Funkce provádí pouze počáteční nastavení vyhledávacího algoritmu.

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	FamilyTyp	INT	Argument pro family kód 1-Wire zařízení.

Tab.VI.11: Funkce OWFamilySearch

Syntaxe: INT OWFamilySearch(INT FamilyTyp, Struct RS232* Interface);

Příklad syntaxe:

```
#include <TempRSWire>;

_GLOBAL RS232 RSPort;

OWFamilySearch(0x28, &RSPort);
```

Funkce OWSerialNum

Funkce pro práci s vyrovnávacím bufferem, který obsahuje sériové číslo (ROM kód). Návratová hodnota je 0.

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN/OUT	SerialNumBuff	USINT	Argument pro přístup pole ROM kódů do bufferu sériového čísla.
IN	DoRead	INT	Argument pro povolení čtení dat z bufferu sériového čísla (čtení ROM kódu = 1, zápis ROM kódu = 0).

Tab.VI.12: Funkce OWSerialNum

Syntaxe: INT OWSerialNum (INT DoRead, USINT* SerialNumBuff, Struct RS232* Interface);

Příklad syntaxe:

```
#include <TempRSWire>;

_GLOBAL USINT ROMCode[8];
_GLOBAL RS232 RSPort;

OWSerialNum(1, ROMCode, &RSPort);
```

Funkce bitacc

Funkce pro přepočet bytů na sériové číslo (ROM kód) a naopak. Návratová hodnota je 1 pokud se zapisuje do bufferu argumentu. Pokud se z bufferu argumentu čte, je návratová hodnota rovná výsledku výpočtu.

I/O	Argument	Datový typ	Popis
IN/OUT	Buff	USINT	Argument bufferu pro přepočet.
IN	Op	INT	Argument pro povolení zápisu výsledku přepočtu zpět do bufferu (zápis do bufferu povolen = 1, zápis do bufferu zakázán = 0).
IN	State	INT	Argument pro volbu typu přepočtu (OR = 1, NAND = 0).
IN	Loc	INT	Argument pro určení pozice dat v bufferu.

Tab.VI.13: Funkce bitacc

Syntaxe: INT bitacc(INT Op, INT State, INT Loc, USINT* Buff);

Funkce OWReset

Funkce pro reset 1-Wire sítě. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.14: Funkce OWReset

Syntaxe: INT OWReset(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
        if(OWDetect("IF1", &RSPort) == 0){
            Test = 1;
        }
}
if(Test == 1){
        Error = OWReset(&RSPort);
        if(Error == 2){
            Test = 1;
        } else{
            Test = 2;
        }
}
```

Funkce OWLevel

Funkce pro nastavení napěťové úrovně na 1-Wire síti. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	NewLevel	INT	Argument pro novou úroveň 1-Wire sítě.

Tab.VI.15: Funkce OWLevel

Argument NewLevel může nabývat pouze 4 hodnot (Normal => 0, Overdrive => 1, Strong => 2 a Program => 3).

Syntaxe: INT OWLevel(INT NewLevel, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
```

```
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
        if(OWDetect("IF1", &RSPort) == 0){
            Test = 1;
        }
}
if(Test == 1){
        Error = OWLevel(0, &RSPort);
        if(Error == 2){
            Test = 1;
        } else{
            Test = 2;
        }
}
```

Funkce OWNext

Funkce s vyhledávací rutinou pro vyhledávání nových 1-Wire zařízení pomocí převodníku DS2480. Funkce je součástí funkce OWFindDevice. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	DoReset	INT	Argument pro povolení resetu 1-Wire sítě na začátku procesu hledání (Reset povolen => 1, Reset zakázán => 0).
IN	AlarmOnly	INT	Argument pro určení typu hledání.

Tab.VI.16: Funkce OWNext

Argument AlarmOnly může nabývat pouze hodnoty 1 nebo 0 (Hledání alarmů na 1-Wire síti => 1, Hledání zařízení na 1-Wire síti => 1).

Syntaxe: `INT OWNext(INT DoReset, INT AlarmOnly, Struct RS232* Interface);`

Funkce OWConvert

Funkce pro konverzi teploty a dat (Přenos dat ze scratchpad paměti do EEPROM paměti). Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	SendByte	INT	Argument pro příkazový byte.

Tab.VI.17: Funkce OWConvert

Příkazový byte slouží pro určení typu konverze (0x44 => Konverze teploty, 0x48 => Přenos dat ze scratchpad paměti do EEPROM paměti).

Syntaxe: `INT OWConvert(INT SendByte, Struct RS232* Interface);`

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWConvert (0x44, &RSPort);
    if(Error == 2){
        Test = 1;
    } else{
        Test = 2;
    }
}
```

Funkce OWMATCHROM

Funkce pro odeslání MATCH ROM příkazu a ROM kódu na 1-Wire síť. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.18: Funkce OWMATCHROM

Syntaxe: INT OWMATCHROM(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
    if(Error == 0){
```

```
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
    }
    if(Test == 2){
        Error = OWMATCHROM(&RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
    }
```

Funkce OWSkipMATCHROM

Funkce pro odeslání Skip MATCH ROM příkazu na 1-Wire síť. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.19: Funkce OWSkipMATCHROM

Syntaxe: INT OWSkipMATCHROM(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
    if(Error == 0){
        Test = 2;
    }
    if(Error == 1){
        Test = 0;
    }
}
if(Test == 2){
```

```
        Error = OWSkipMATCHROM(&RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
}
```

Funkce OWWriteConfig

Funkce pro odeslání konfiguračních dat na 1-Wire zařízení dané ROM kódem. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	ROMCode	USINT	Argument pro ROM kód 1-Wire zařízení.
IN	AlarmTH	USINT	Argument pro horní teplotní byte alarmové hodnoty.
IN	AlarmTL	USINT	Argument pro dolní teplotní byte alarmové hodnoty.
IN	TempConfig	USINT	Argument pro konfigurační byte teploty.

Tab.VI.20: Funkce OWWriteConfig

Konfigurační byte teploty může nabývat pouze 4 hodnot (Rozsah 12 bitů => 0x7F, Rozsah 11 bitů => 0x5F, Rozsah 10 bitů => 0x3F a Rozsah 9 bitů => 0x1F). Byte alarmové hodnoty má dvě části. Dolní 4 bity jsou vždy v logické jedničce (např. 0x1F, 0xAF).

Syntaxe: INT OWWriteConfig(USINT AlarmTH, USINT AlarmTL, USINT TempConfig, USINT* ROMCode, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
    if(Error == 0){
        Test = 2;
    }
```

```
        if(Error == 1){
            Test = 0;
        }
    }
    if(Test == 2){
        Error = OWWriteConfig(0xAF, 0x1F, 0x7F, &ROMCode[0], &RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
    }
```

Funkce OWWriteConfigAll

Funkce pro odeslání konfiguračních dat na všechna zařízení umístěných na 1-Wire síti. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	ConDelay	UINT	Argument pro realizaci zpoždění při zápisu konfiguračních dat.
IN	AlarmTH	USINT	Argument pro horní teplotní byte alarmové hodnoty.
IN	AlarmTL	USINT	Argument pro dolní teplotní byte alarmové hodnoty.
IN	TempConfig	USINT	Argument pro konfigurační byte teploty.

Tab.VI.21: Funkce OWWriteConfigAll

Konfigurační byte teploty může nabývat pouze 4 hodnot (Rozsah 12 bitů => 0x7F, Rozsah 11 bitů => 0x5F, Rozsah 10 bitů => 0x3F a Rozsah 9 bitů => 0x1F). Byte alarmové hodnoty má dvě části. Dolní 4 bity jsou vždy v logické jedničce (např. 0x1F, 0xAF). Hodnota argumentu ConDelay je závislá na použitém tasku. Doba zpoždění musí být minimálně 1 sekundu.

Syntaxe: INT OWWriteConfigAll(USINT AlarmTH, USINT AlarmTL, USINT TempConfig, UINT ConDelay, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL UINT Time = 10;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
```

```
            Test = 1;
        }
    }
    if(Test == 1){
        Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
        if(Error == 0){
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
    }
    if(Test == 2){
        Error = OWWriteConfigAll(0xAF, 0x1F, 0x7F,  Time, &RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
    }
```

Funkce OWReadScratchpad

Funkce pro čtení scratchpad paměti z 1-Wire zařízení dané ROM kódem. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	ROMCode	USINT	Argument pro ROM kód 1-Wire zařízení.
OUT	Scratchpad	USINT	Argument pro data získané z 1-Wire zařízení.

Tab.VI.22: Funkce OWReadScratchpad

Syntaxe: INT OWReadScratchpad(USINT* ROMCode, USINT* Scratchpad, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL USINT Scratchpad[9];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
```

```
}
if(Test == 1){
        Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
        if(Error == 0){
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
}
if(Test == 2){
        Error = OWReadScratchpad(&ROMCode[0], &Scratchpad, &RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
}
```

Funkce OWCopyScratchpad

Funkce pro kopírování dat ze scratchpad paměti do EEPROM paměti. U 1-Wire zařízení daného ROM kódem. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	ROMCode	USINT	Argument pro ROM kód 1-Wire zařízení.

Tab.VI.23: Funkce OWCopyScratchpad

Syntaxe: INT OWCopyScratchpad(USINT* ROMCode, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
        if(OWDetect("IF1", &RSPort) == 0){
            Test = 1;
        }
}
if(Test == 1){
        Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
        if(Error == 0){
            Test = 2;
```

```
        }
        if(Error == 1){
            Test = 0;
        }
    }
    if(Test == 2){
        Error = OWCopyScratchpad(&ROMCode[0], &RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
    }
```

Funkce OWCopyScratchpadAll

Funkce pro kopírování dat ze scratchpad paměti do EEPROM paměti. Pro všechny zařízení na 1-Wire síti. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.

Tab.VI.24: Funkce OWCopyScratchpadAll

Syntaxe: INT OWCopyScratchpadAll(Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
    if(Error == 0){
        Test = 2;
    }
    if(Error == 1){
        Test = 0;
    }
}
```

```
if(Test == 2){
        Error = OWCopyScratchpadAll(&RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
}
```

Funkce OWPresentDevice

Funkce provádějící test přítomnosti zařízení na 1-Wire síti. Test se provádí na základě odeslání ROM kódu v podobě vyhledávacího algoritmu. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	AlarmOnly	INT	Argument pro určení typu hledání.

Tab.VI.25: Funkce OWPresentDevice

Argument AlarmOnly může nabývat pouze hodnoty 1 nebo 0 (Hledání alarmů na 1-Wire síti => 1, Hledání zařízení na 1-Wire síti => 1).

Syntaxe: INT OWPresentDevice(INT AlarmOnly, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
        if(OWDetect("IF1", &RSPort) == 0){
            Test = 1;
        }
}
if(Test == 1){
        Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
        if(Error == 0){
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
}
if(Test == 2){
        Error = OWPresentDevice(1, &RSPort);
```

```
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
}
```

Funkce OWReadTemp

Funkce pro čtení teploty z 1-Wire zařízení. 1-Wire zařízení se vybírá na základě ROM kódu. Návratová hodnota je 0, 1 nebo 2. Jedná se o chybový příznak funkce (0 => bez chyby; 1 => chyba; 2 => opakovat funkci).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN	SkipROM	UINT	Argument pro povolení Skip MATCH ROM příkazu (Skip MATCH ROM => 1, MATCH ROM => 0).
IN	ConDelay	UINT	Argument pro realizaci zpoždění při konverzi teploty na 1-Wire zařízení.
IN	ROMCode	USINT	Argument pro ROM kód 1-Wire zařízení.
OUT	Temp	REAL	Argument pro získanou teplotu 1-Wire zařízení.

Tab.VI.26: Funkce OWReadTemp

Hodnota argumentu ConDelay je závislá na použitém tasku a na konfiguraci 1-Wire zařízení (Rozlišení 12 bitů => minimálně 750ms, rozlišení 11 bitů => minimálně 375ms, rozlišení 10 bitů => minimálně 188ms a rozlišení 9 bitů => minimálně 94ms).

Syntaxe: INT OWReadTemp(UINT SkipROM, UINT ConDelay, USINT* ROMCode, REAL* Temp, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL INT FindDevice;
_GLOBAL UINT Time = 10;
_GLOBAL REAL Temp;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
    if(OWDetect("IF1", &RSPort) == 0){
        Test = 1;
    }
}
if(Test == 1){
    Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
```

```
        if(Error == 0){
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
    }
    if(Test == 2){
        Error = OWReadTemp(1, Time, &ROMCode[0], &Temp, &RSPort);
        if(Error == 2){
            Test = 2;
        } else{
            Test = 3;
        }
    }
```

Funkce OWCalculationTemp

Funkce pro přepočet teploty. Návratová hodnota je rovna přepočítané teplotě. V případě chyby se vrací hodnota 3276,6 (Chybová hodnota).

I/O	Argument	Datový typ	Popis
IN	ScratchpadTemp	USINT	Argument pro data scratchpad paměti 1-Wire zařízení.

Tab.VI.27: Funkce OWCalculationTemp

Syntaxe: `INT OWCalculatonTemp(USINT* Scratchpad);`

Funkce OWMeasureTemp

Funkce řešící celý proces měření a vyhledávání zařízení na 1-Wire síti. Návratová hodnota je 0, 1, 2, 3, 4 nebo 5. Jedná se o chybový příznak funkce (0 => chyba; 1 => opakovat funkci; 2 => vyhledávání dokončeno; 3 => 1-Wire je aktivní, ale neměří se; 4 => konec procesu měření; 5 => opakovat proces měření).

I/O	Argument	Datový typ	Popis
IN/OUT	Interface	RS232	Argument pro strukturu RS232.
IN/OUT	EnableConfig	UINT	Argument pro povolení konfigurace zařízení 1-Wire sítě (Konfigurace povolena => 1, Konfigurace zakázána => 0).
IN/OUT	ResetFindDevice	USINT	Argument pro vynulování vyhledávacích parametrů.
IN	Family	INT	Argument pro family kód 1-Wire zařízení. (28h => DS18B20, 10h => DS1920 a DS1820)
IN	MaxDevices	INT	Argument pro požadavek nalezení maximálního počtu 1-Wire zařízení.
IN	AlarmTH	USINT	Argument pro horní teplotní byte alarmové hodnoty.
IN	AlarmTL	USINT	Argument pro dolní teplotní byte alarmové hodnoty.

IN	TempConfig	USINT	Argument pro konfigurační byte teploty.
IN	SkipROM	UINT	Argument pro povolení Skip MATCH ROM příkazu (Skip MATCH ROM => 1, MATCH ROM => 0).
IN	ConvertDelay	UINT	Argument pro realizaci zpoždění při konverzi teploty a dat na 1-Wire zařízení.
IN	EnableReadTemp	UINT	Argument pro povolení procesu měření (Měření povoleno => 1, Měření zakázáno => 0).
IN	PortDevice	USINT	Argument pro nastavení označení IF portu.
OUT	MTemp	REAL	Argument pro získanou teplotu 1-Wire zařízení.
OUT	NumberDevice	USINT	Argument pro počet nalezených 1-Wire zařízení.
OUT	ROMCode	USINT[8]	Argument pole pro ROM kód nalezeného 1-Wire zařízení.

Tab.VI.28: Funkce OWMeasureTemp

Hodnota argumentu ConvertDelay je závislá na použitém tasku a na konfiguraci 1-Wire zařízení (Rozlišení 12 bitů => minimálně 750ms, rozlišení 11 bitů => minimálně 375ms, rozlišení 10 bitů => minimálně 188ms a rozlišení 9 bitů => minimálně 94ms). Konfigurační byte teploty může nabývat pouze 4 hodnot (Rozsah 12 bitů => 0x7F, Rozsah 11 bitů => 0x5F, Rozsah 10 bitů => 0x3F a Rozsah 9 bitů => 0x1F). Byte alarmové hodnoty má dvě části. Dolní 4 bity jsou vždy v logické jedničce (např. 0x1F, 0xAF).

Syntaxe: INT OWMeaureTemp(USINT ROMCode[8], INT Family, INT MaxDevices, USINT AlarmTH, USINT AlarmTL, USINT TempConfig, UINT SkipROM, UINT ConvertDelay, UINT EnableReadTemp, UINT* EnableConfig, USINT* PortDevice, REAL* MTemp, USINT* NumberDevice, USINT* ResetFindDevice, Struct RS232* Interface);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;

_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL USINT FindDevice;
_GLOBAL USINT ResetFindParameter = 1;
_GLOBAL UINT Time = 10;
_GLOBAL UINT Config = 1;
_GLOBAL REAL Temp[30];
_GLOBAL INT Error;

Error = OWMeaureTemp(&ROMCode[RSPort.NumDev1], 0x28, 30, 0x7F, 0x1F, 0x7F, 1, Time, 1, &Config, “IF1”, &Temp[RSPort.NumDev2], &FindDevice, &ResetFindParameter, &RSPort);
```

Funkce msDelay

Funkce pro realizaci krátkého zpoždění. Návratová hodnota je 0. Tuto funkci je možné použít pro realizaci zpoždění maximálně jednotek milisekund. Doba zpoždění je závislá na době tasku. V případě přetečení času (čas zpoždění je větší než čas tasku).

I/O	Argument	Datový typ	Popis
IN	j	INT	Argument pro požadovaný čas zpoždění (milisekundy).

Tab.VI.29: Funkce msDelay

Syntaxe: INT msDelay(INT j);

Příklad syntaxe:

```
#include <TempRSWire>;

_GLOBAL DINT Time;

msDelay(10);
Time++;
if(Time == 100000){
        Time = 0;
}
```

Funkce HexToChar

Funkce pro převod 8 bitového hexadecimálního čísla na dva znaky. Návratová hodnota je 0.

I/O	Argument	Datový typ	Popis
IN	Hex	USINT	Argument s požadovanou hodnotou pro převod.
OUT	Char1	USINT	Argument s prvním převedeným znakem.
OUT	Char2	USINT	Argument s druhým převedeným znakem.

Tab.VI.30: Funkce HexToChar

Syntaxe: INT HexToChar(USINT Hex, USINT* Char1, USINT* Char2);

Příklad syntaxe:

```
#include <TempRSWire>;

_GLOBAL USINT HexNumber;
_GLOBAL USINT Char1;
_GLOBAL USINT Char2;

HexToChar(HexNumber, &Char1, &Char2);
```

Funkce SetCRC8

Funkce pro vynulování výsledku výpočtu 8 bitového CRC kódu a inicializaci tabulky 8 bitových CRC hodnot převodníku DS2480. Návratová hodnota je 0.

I/O	Argument	Datový typ	Popis
IN	Reset	USINT	Argument pro počáteční nastavení CRC hodnoty.

Tab.VI.31: Funkce SetCRC8

Syntaxe: INT SetCRC8(USINT Reset);

Příklad syntaxe:

```
#include <TempRSWire>;
SetCRC8(0);
```

Funkce ResetTableCRC8

Funkce obsahuje tabulku 8 bitových CRC hodnot převodníku DS2480. Návratová hodnota je 0.

Syntaxe: INT ResetTableCRC8(void);

Funkce SetCRC8

Funkce vrací hodnotu proměnné PowerUp (indikuje přítomnost převodníku ADA-101W). Návratová hodnota je rovná hodnotě proměnné PowerUp.

Syntaxe: USINT ReadPowerUp(void);

Příklad syntaxe:

```
#include <TempRSWire>;
_GLOBAL USINT PowerUp;
PowerUp = ReadPowerUp();
```

Funkce DoCRC8

Funkce pro výpočet 8 bitových CRC hodnot. Návratová hodnota je výsledek výpočtu 8 bitového CRC kódu.

I/O	Argument	Datový typ	Popis
IN	x	USINT	Argument s požadovanou hodnotou pro výpočet 8 bitového CRC kódu.

Tab.VI.32: Funkce DoCRC8

Syntaxe: USINT DoCRC8(USINT x);

Příklad syntaxe:

```
#include <dvframe>;
#include <TempRSWire>;
_GLOBAL RS232 RSPort;
_GLOBAL USINT ROMCode[30][8];
_GLOBAL USINT Scratchpad[9];
_GLOBAL USINT CRC;
```

```
_GLOBAL INT FindDevice;
_GLOBAL INT Test;
_GLOBAL INT Error;

if(Test == 0){
        if(OWDetect("IF1", &RSPort) == 0){
            Test = 1;
        }
}
if(Test == 1){
        Error = OWFindDevices(0x28, 30, ROMCode[FindDevice], &FindDevice, &RSPort);
        if(Error == 0){
            Test = 2;
        }
        if(Error == 1){
            Test = 0;
        }
}
if(Test == 2){
        Error = OWReadScratchpad(&ROMCode[0], &Scratchpad, &RSPort);
        if(Error == 0){
            for(INT i = 0; i < 9; i++){
                CRC = DoCRC8(Scratchpad[i]);
            }
            Test = 3;
        }
        if(Error == 1){
            Test = 0;
        }
        if(Error == 2){
            Test = 2;
        }
}
```

PŘÍLOHA VII – Elektronická příloha (DVD)

DVD obsahuje:

1. Aplikace společnosti CEL-MAR - Tempf.exe
2. Visual Studio - RSWire (Testovací aplikace RSWire)
3. Knihovny - ASTime (Knihovna pro práci s informacemi času a data)
   - DVFrame (Knihovna pro ovládání sériového portu)
   - TempRSWire (Realizovaná knihovna)
4. Automation Studio - Laboratorní aplikace (Testovací aplikace pro PLC automat)
   - Vrty (Aplikace pro reálný provoz)
   - PVIOPCConfiguratorFinal.mdb (Nastavení PVI OPC konfigurátoru)
5. Promotic - Laboratorní aplikace (Vizualizace testovací aplikace)
   - Pozorovací vrty (Vizualizace reálného provozu)
6. Obrázky - Diagramy funkcí, fotky, obrázky vizualizace a ASCII tabulka
7. Dokumenty - ADA-101W User Manual.pdf
   - DS18B20 Datasheet.pdf
   - DS18S20 Datasheet.pdf
   - DS1920 Datasheet.pdf
   - DS2480B Datasheet.pdf
   - Logické systémy řízení skripta.pdf
   - ROM kódy čidel pozorovacího vrtu.pdf
   - Semestrální práce – Musiol a Černoch.pdf
   - X20 User Manual.pdf
8. Závěrečná zpráva diplomové práce - Diplomová práce Tomáš Vavrla.pdf